Formal Modelling, Analysis and Verification of Hybrid Systems

Naijun Zhan

State Key Lab. of Comp. Sci., Institute of Software, Chinese Academy of Sciences

Mini Course on HSs

Beijing, Dec. 4-25

Outline

Hybrid systems

Definition

Hybrid systems exhibit combinations of discrete jumps and continuous evolution.

Examples

High-speed train control systems (ETCS, CTCS), air traffic control systems, nuclear reaction control systems, aircraft control systems, spacecraft control systems, ...

Features

- **Interaction between discrete and continuous evolution**;
- **Safety-critical**;
- **Interdiscipline**.

Hybrid systems

Definition

Hybrid systems exhibit combinations of discrete jumps and continuous evolution.

Examples

High-speed train control systems (ETCS, CTCS), air traffic control systems, nuclear reaction control systems, aircraft control systems, spacecraft control systems, ...

Features

- **Interaction between discrete and continuous evolution**;
- **Safety-critical**;
- **Interdiscipline**.

Hybrid systems

Definition

Hybrid systems exhibit combinations of discrete jumps and continuous evolution.

Examples

High-speed train control systems (ETCS, CTCS), air traffic control systems, nuclear reaction control systems, aircraft control systems, spacecraft control systems, ...

Features

- **Interaction between discrete and continuous evolution**;
- **Safety-critical**;
- **Interdiscipline**.

Issues of hybrid systems

- **Modelling:**
  - To establish a model for the system to be developed with precise mathematical semantics
  - Have to consider: **concurrency, deterministic vs nondeterministic, continuous vs discrete, communication, static vs dynamic (mobility, adaptability), qualitative vs quantitative (predicability), real-time**, ...
- **Simulation:**
  - To obtain a possible execution of the model upto a finite time horizon using numerical methods
  - Well accepted in industrial practice
- **Verification:**
  - Using mathematical approach to prove if a model satisfies the desired properties (specification)
  - Main methods include: **model-checking, theorem proving, abstract interpretation**

Issues of hybrid systems

- **Modelling:**
  - To establish a model for the system to be developed with precise mathematical semantics
  - Have to consider: **concurrency, deterministic vs nondeterministic, continuous vs discrete, communication, static vs dynamic (mobility, adaptability), qualitative vs quantitative (predicability), real-time**, ...
- **Simulation:**
  - To obtain a possible execution of the model upto a finite time horizon using numerical methods
  - Well accepted in industrial practice
- **Verification:**
  - Using mathematical approach to prove if a model satisfies the desired properties (specification)
  - Main methods include: **model-checking, theorem proving, abstract interpretation**

Issues of hybrid systems

- **Modelling:**
  - To establish a model for the system to be developed with precise mathematical semantics
  - Have to consider: **concurrency, deterministic vs nondeterministic, continuous vs discrete, communication, static vs dynamic (mobility, adaptability), qualitative vs quantitative (predicability), real-time**, ...
- **Simulation:**
  - To obtain a possible execution of the model upto a finite time horizon using numerical methods
  - Well accepted in industrial practice
- **Verification:**
  - Using mathematical approach to prove if a model satisfies the desired properties (specification)
  - Main methods include: **model-checking, theorem proving, abstract interpretation**

Issues of hybrid systems

- **Modelling:**
  - To establish a model for the system to be developed with precise mathematical semantics
  - Have to consider: **concurrency, deterministic vs nondeterministic, continuous vs discrete, communication, static vs dynamic (mobility, adaptability), qualitative vs quantitative (predicability), real-time**, ...
- **Simulation:**
  - To obtain a possible execution of the model upto a finite time horizon using numerical methods
  - Well accepted in industrial practice
- **Verification:**
  - Using mathematical approach to prove if a model satisfies the desired properties (specification)
  - Main methods include: **model-checking, theorem proving, abstract interpretation**

Issues of hybrid systems

- **Modelling:**
  - To establish a model for the system to be developed with precise mathematical semantics
  - Have to consider: **concurrency, deterministic vs nondeterministic, continuous vs discrete, communication, static vs dynamic (mobility, adaptability), qualitative vs quantitative (predicability), real-time**, ...
- **Simulation:**
  - To obtain a possible execution of the model upto a finite time horizon using numerical methods
  - Well accepted in industrial practice
- **Verification:**
  - Using mathematical approach to prove if a model satisfies the desired properties (specification)
  - Main methods include: **model-checking, theorem proving, abstract interpretation**

Issues of hybrid systems

- **Modelling:**
  - To establish a model for the system to be developed with precise mathematical semantics
  - Have to consider: **concurrency, deterministic vs nondeterministic, continuous vs discrete, communication, static vs dynamic (mobility, adaptability), qualitative vs quantitative (predicability), real-time**, ...
- **Simulation:**
  - To obtain a possible execution of the model upto a finite time horizon using numerical methods
  - Well accepted in industrial practice
- **Verification:**
  - Using mathematical approach to prove if a model satisfies the desired properties (specification)
  - Main methods include: **model-checking, theorem proving, abstract interpretation**

Issues of hybrid systems

- **Modelling:**
  - To establish a model for the system to be developed with precise mathematical semantics
  - Have to consider: **concurrency, deterministic vs nondeterministic, continuous vs discrete, communication, static vs dynamic (mobility, adaptability), qualitative vs quantitative (predicability), real-time**, ...
- **Simulation:**
  - To obtain a possible execution of the model upto a finite time horizon using numerical methods
  - Well accepted in industrial practice
- **Verification:**
  - Using mathematical approach to prove if a model satisfies the desired properties (specification)
  - Main methods include: **model-checking, theorem proving, abstract interpretation**

Issues of hybrid systems

- **Modelling:**
  - To establish a model for the system to be developed with precise mathematical semantics
  - Have to consider: **concurrency, deterministic vs nondeterministic, continuous vs discrete, communication, static vs dynamic (mobility, adaptability), qualitative vs quantitative (predicability), real-time**, ...
- **Simulation:**
  - To obtain a possible execution of the model upto a finite time horizon using numerical methods
  - Well accepted in industrial practice
- **Verification:**
  - Using mathematical approach to prove if a model satisfies the desired properties (specification)
  - Main methods include: **model-checking, theorem proving, abstract interpretation**

Issues of hybrid systems

- **Modelling:**
  - To establish a model for the system to be developed with precise mathematical semantics
  - Have to consider: **concurrency, deterministic vs nondeterministic, continuous vs discrete, communication, static vs dynamic (mobility, adaptability), qualitative vs quantitative (predicability), real-time**, ...
- **Simulation:**
  - To obtain a possible execution of the model upto a finite time horizon using numerical methods
  - Well accepted in industrial practice
- **Verification:**
  - Using mathematical approach to prove if a model satisfies the desired properties (specification)
  - Main methods include: **model-checking, theorem proving, abstract interpretation**

Issues of hybrid systems (Cont'd)

- **Synthesis:** The process of computing an implementation (the "how") from a specification of the desired behavior and performance (the "what") and the assumptions on the environment (the "where")
- **Qualitative issues:**
  - Total absence of undesirable behavior is an overly ambitious goal, being economically unattainable or even technically impossible due to
    - uncontrollable environment influences;
    - unavoidable manufacturing tolerance;
    - component breakdown, etc.
  - The existing qualitative safety analysis methods for hybrid systems have to be complemented quantitative methods, quantifying the likelihood of residual errors or the related performance figures in systems subject to uncertain, stochastic behavior as well as noise.
- **Other issues:** stability, controllability, observability, . . .

Issues of hybrid systems (Cont'd)

- **Synthesis:** The process of computing an implementation (the "how") from a specification of the desired behavior and performance (the "what") and the assumptions on the environment (the "where")
- **Qualitative issues:**
  - Total absence of undesirable behavior is an overly ambitious goal, being economically unattainable or even technically impossible due to
    - uncontrollable environment influences;
    - unavoidable manufacturing tolerance;
    - component breakdown, etc.
  - The existing qualitative safety analysis methods for hybrid systems have to be complemented quantitative methods, quantifying the likelihood of residual errors or the related performance figures in systems subject to uncertain, stochastic behavior as well as noise.
- **Other issues:** stability, controllability, observability, ...

Issues of hybrid systems (Cont'd)

- **Synthesis:** The process of computing an implementation (the "how") from a specification of the desired behavior and performance (the "what") and the assumptions on the environment (the "where")
- **Qualitative issues:**
    - Total absence of undesirable behavior is an overly ambitious goal, being economically unattainable or even technically impossible due to
        - uncontrollable environment influences;
        - unavoidable manufacturing tolerance;
        - component breakdown, etc.
    - The existing qualitative safety analysis methods for hybrid systems have to be complemented quantitative methods, quantifying the likelihood of residual errors or the related performance figures in systems subject to uncertain, stochastic behavior as well as noise.
- **Other issues:** stability, controllability, observability, ...

Issues of hybrid systems (Cont'd)

- **Synthesis:** The process of computing an implementation (the "how") from a specification of the desired behavior and performance (the "what") and the assumptions on the environment (the "where")
- **Qualitative issues:**
  - Total absence of undesirable behavior is an overly ambitious goal, being economically unattainable or even technically impossible due to
    - uncontrollable environment influences;
    - unavoidable manufacturing tolerance;
    - component breakdown, etc.
  - The existing qualitative safety analysis methods for hybrid systems have to be complemented quantitative methods, quantifying the likelihood of residual errors or the related performance figures in systems subject to uncertain, stochastic behavior as well as noise.
- **Other issues:** stability, controllability, observability, . . .

Issues of hybrid systems (Cont'd)

- **Synthesis:** The process of computing an implementation (the "how") from a specification of the desired behavior and performance (the "what") and the assumptions on the environment (the "where")
- **Qualitative issues:**
    - Total absence of undesirable behavior is an overly ambitious goal, being economically unattainable or even technically impossible due to
        - uncontrollable environment influences;
        - unavoidable manufacturing tolerance;
        - component breakdown, etc.
    - The existing qualitative safety analysis methods for hybrid systems have to be complemented quantitative methods, quantifying the likelihood of residual errors or the related performance figures in systems subject to uncertain, stochastic behavior as well as noise.
- **Other issues:** stability, controllability, observability, ...

Issues of hybrid systems (Cont'd)

- **Synthesis:** The process of computing an implementation (the "how") from a specification of the desired behavior and performance (the "what") and the assumptions on the environment (the "where")
- **Qualitative issues:**
  - Total absence of undesirable behavior is an overly ambitious goal, being economically unattainable or even technically impossible due to
    - uncontrollable environment influences;
    - unavoidable manufacturing tolerance;
    - component breakdown, etc.
  - The existing qualitative safety analysis methods for hybrid systems have to be complemented quantitative methods, quantifying the likelihood of residual errors or the related performance figures in systems subject to uncertain, stochastic behavior as well as noise.
- **Other issues:** stability, controllability, observability, ...

Issues of hybrid systems (Cont'd)

- **Synthesis:** The process of computing an implementation (the "how") from a specification of the desired behavior and performance (the "what") and the assumptions on the environment (the "where")
- **Qualitative issues:**
  - Total absence of undesirable behavior is an overly ambitious goal, being economically unattainable or even technically impossible due to
    - uncontrollable environment influences;
    - unavoidable manufacturing tolerance;
    - component breakdown, etc.
  - The existing qualitative safety analysis methods for hybrid systems have to be complemented quantitative methods, quantifying the likelihood of residual errors or the related performance figures in systems subject to uncertain, stochastic behavior as well as noise.
- **Other issues:** stability, controllability, observability, ...

Issues of hybrid systems (Cont'd)

- **Synthesis:** The process of computing an implementation (the "how") from a specification of the desired behavior and performance (the "what") and the assumptions on the environment (the "where")
- **Qualitative issues:**
  - Total absence of undesirable behavior is an overly ambitious goal, being economically unattainable or even technically impossible due to
    - uncontrollable environment influences;
    - unavoidable manufacturing tolerance;
    - component breakdown, etc.
  - The existing qualitative safety analysis methods for hybrid systems have to be complemented quantitative methods, quantifying the likelihood of residual errors or the related performance figures in systems subject to uncertain, stochastic behavior as well as noise.
- **Other issues:** **stability, controllability, observability,** ...

Related Work

Automata-based techniques

- **Modeling: Phase transition systems** [Manna&Pnueli,1993] **Hybrid automata** [Alur et al, 1995]
  - **Advantages:** intuitive, easy to model the behavior of systems, the basis for model-checking.
  - **Disadvantages:** lacks of structured information, not easy to model complex system.
- **Verification by computing reachable set: model-checking** [Alur et al, 1995], **decision procedure** [LPY, 2001],
  - **Basic idea:** partitioning infinite state space into finite many equivalent classes according to the solution of ODEs, or representing by O-minimal structures
  - **Advantages:** automatic
  - **Disadvantages:** cannot scale up
  - **Focuses:** **symbolic computation, abstraction, approximation**

Related Work

Automata-based techniques

- **Modeling: Phase transition systems** [Manna&Pnueli,1993] **Hybrid automata** [Alur et al, 1995]
  - **Advantages:** intuitive, easy to model the behavior of systems, the basis for model-checking.
  - **Disadvantages:** lacks of structured information, not easy to model complex system.
- **Verification by computing reachable set: model-checking** [Alur et al, 1995], **decision procedure** [LPY, 2001],
  - **Basic idea:** partitioning infinite state space into finite many equivalent classes according to the solution of ODEs, or representing by O-minimal structures
  - **Advantages:** automatic
  - **Disadvantages:** cannot scale up
  - **Focuses: symbolic computation, abstraction, approximation**

Related Work

Automata-based techniques

- **Modeling: Phase transition systems** [Manna&Pnueli,1993] **Hybrid automata** [Alur et al, 1995]
  - **Advantages:** intuitive, easy to model the behavior of systems, the basis for model-checking.
  - **Disadvantages:** lacks of structured information, not easy to model complex system.
- **Verification by computing reachable set: model-checking** [Alur et al, 1995], **decision procedure** [LPY, 2001],
  - **Basic idea:** partitioning infinite state space into finite many equivalent classes according to the solution of ODEs, or representing by O-minimal structures
  - **Advantages:** automatic
  - **Disadvantages:** cannot scale up
  - **Focuses: symbolic computation, abstraction, approximation**

Related Work

Automata-based techniques

- **Modeling: Phase transition systems** [Manna&Pnueli,1993] **Hybrid automata** [Alur et al, 1995]
  - **Advantages:** intuitive, easy to model the behavior of systems, the basis for model-checking.
  - **Disadvantages:** lacks of structured information, not easy to model complex system.
- **Verification by computing reachable set: model-checking** [Alur et al, 1995], **decision procedure** [LPY, 2001],
  - **Basic idea:** partitioning infinite state space into finite many equivalent classes according to the solution of ODEs, or representing by O-minimal structures
  - **Advantages:** automatic
  - **Disadvantages:** cannot scale up
  - **Focuses:** **symbolic computation, abstraction, approximation**

Related Work

Automata-based techniques

- **Modeling: Phase transition systems** [Manna&Pnueli,1993] **Hybrid automata** [Alur et al, 1995]
  - **Advantages:** intuitive, easy to model the behavior of systems, the basis for model-checking.
  - **Disadvantages:** lacks of structured information, not easy to model complex system.
- **Verification by computing reachable set: model-checking** [Alur et al, 1995], **decision procedure** [LPY, 2001],
  - **Basic idea:** partitioning infinite state space into finite many equivalent classes according to the solution of ODEs, or representing by O-minimal structures
  - **Advantages:** automatic
  - **Disadvantages:** cannot scale up
  - **Focuses:** symbolic computation, abstraction, approximation

Related Work

Automata-based techniques

- **Modeling: Phase transition systems** [Manna&Pnueli,1993] **Hybrid automata** [Alur et al, 1995]
  - **Advantages:** intuitive, easy to model the behavior of systems, the basis for model-checking.
  - **Disadvantages:** lacks of structured information, not easy to model complex system.
- **Verification by computing reachable set: model-checking** [Alur et al, 1995], **decision procedure** [LPY, 2001],
  - **Basic idea:** partitioning infinite state space into finite many equivalent classes according to the solution of ODEs, or representing by O-minimal structures
  - **Advantages:** automatic
  - **Disadvantages:** cannot scale up
  - **Focuses: symbolic computation, abstraction, approximation**

Related Work

Automata-based techniques

- **Modeling: Phase transition systems** [Manna&Pnueli,1993] **Hybrid automata** [Alur et al, 1995]
  - **Advantages:** intuitive, easy to model the behavior of systems, the basis for model-checking.
  - **Disadvantages:** lacks of structured information, not easy to model complex system.
- **Verification by computing reachable set: model-checking** [Alur et al, 1995], **decision procedure** [LPY, 2001],
  - **Basic idea:** partitioning infinite state space into finite many equivalent classes according to the solution of ODEs, or representing by O-minimal structures
  - **Advantages:** automatic
  - **Disadvantages:** cannot scale up
  - **Focuses:** symbolic computation, abstraction, approximation

Related Work

Automata-based techniques

- **Modeling: Phase transition systems** [Manna&Pnueli,1993] **Hybrid automata** [Alur et al, 1995]
    - **Advantages:** intuitive, easy to model the behavior of systems, the basis for model-checking.
    - **Disadvantages:** lacks of structured information, not easy to model complex system.
- **Verification by computing reachable set: model-checking** [Alur et al, 1995], **decision procedure** [LPY, 2001],
    - **Basic idea:** partitioning infinite state space into finite many equivalent classes according to the solution of ODEs, or representing by O-minimal structures
    - **Advantages:** automatic
    - **Disadvantages:** cannot scale up
    - **Focuses: symbolic computation, abstraction, approximation**

Related Work (Cont'd)

Compositional modeling approaches

- **Modeling environment:** **SHIFT** [DGV 1996]
- **Hierarchical modeling:** **PTOLEMY** [Lee et al 2003]
- **Modular modeling:** **I/O hybrid automata** [Lynch et al 1996], **hybrid modules** [Alur et al 2003], **CHARON** [Alur&Henzinger 1997]
- **Algebraic approach:** **Hybrid CSP** [He 1994, Zhou et al 1995]

Problem

It lacks of verification techniques for these compositional modelling techniques

Related Work (Cont'd)

Compositional modeling approaches

- **Modeling environment: SHIFT** [DGV 1996]
- **Hierarchical modeling: PTOLEMY** [Lee et al 2003]
- **Modular modeling: I/O hybrid automata** [Lynch et al 1996], **hybrid modules** [Alur et al 2003], **CHARON** [Alur&Henzinger 1997]
- **Algebraic approach: Hybrid CSP** [He 1994, Zhou et al 1995]

Problem

It lacks of verification techniques for these compositional modelling techniques

Related Work (Cont'd)

Compositional modeling approaches

- **Modeling environment: SHIFT** [DGV 1996]
- **Hierarchical modeling: PTOLEMY** [Lee et al 2003]
- **Modular modeling: I/O hybrid automata** [Lynch et al 1996], **hybrid modules** [Alur et al 2003], **CHARON** [Alur&Henzinger 1997]
- **Algebraic approach: Hybrid CSP** [He 1994, Zhou et al 1995]

Problem

It lacks of verification techniques for these compositional modelling techniques

Related Work (Cont'd)

Compositional modeling approaches

- **Modeling environment: SHIFT** [DGV 1996]
- **Hierarchical modeling: PTOLEMY** [Lee et al 2003]
- **Modular modeling: I/O hybrid automata** [Lynch et al 1996], **hybrid modules** [Alur et al 2003], **CHARON** [Alur&Henzinger 1997]
- **Algebraic approach: Hybrid CSP** [He 1994, Zhou et al 1995]

Problem

It lacks of verification techniques for these compositional modelling techniques

Related Work (Cont'd)

Compositional modeling approaches

- **Modeling environment: SHIFT** [DGV 1996]
- **Hierarchical modeling: PTOLEMY** [Lee et al 2003]
- **Modular modeling: I/O hybrid automata** [Lynch et al 1996], **hybrid modules** [Alur et al 2003], **CHARON** [Alur&Henzinger 1997]
- **Algebraic approach: Hybrid CSP** [He 1994, Zhou et al 1995]

Problem

It lacks of verification techniques for these compositional modelling techniques

Related Work (Cont'd)

Compositional modeling approaches

- **Modeling environment:** **SHIFT** [DGV 1996]
- **Hierarchical modeling:** **PTOLEMY** [Lee et al 2003]
- **Modular modeling:** **I/O hybrid automata** [Lynch et al 1996], **hybrid modules** [Alur et al 2003], **CHARON** [Alur&Henzinger 1997]
- **Algebraic approach:** **Hybrid CSP** [He 1994, Zhou et al 1995]

Problem

It lacks of verification techniques for these compositional modelling techniques

Related work (Cont'd)

Deduction based approach [Platzer&Clarke 2008]

- **Basic idea:** extending **Floyd-Hoare-Naur inductive assertion method** to hybrid systems.
- **Elements:**
  - A compositional modelling laguage
  - A **Hoare logic**-like specification logic
  - Invariant generation
- **Well-known compositional modelling languages:** **hybrid programs** [Platzer&Clarke 2008], **HCSP** [He 1994, Zhou et al 1995], ...
- **Hybrid specification logics:** **DDL** [Platzer2008], **DADL** [Platzer2010], **EDC** [Zhou et al 1994], ...
- **Advantages:** scalability
- **Difficulties:** invariant generation

Related work (Cont'd)

Deduction based approach [Platzer&Clarke 2008]

- **Basic idea:** extending **Floyd-Hoare-Naur inductive assertion method** to hybrid systems.
- **Elements:**
  - A compositional modelling laguage
  - A Hoare logic-like specification logic
  - Invariant generation
- **Well-known compositional modelling languages:** **hybrid programs** [Platzer&Clarke 2008], **HCSP** [He 1994, Zhou et al 1995], ...
- **Hybrid specification logics:** **DDL** [Platzer2008], **DADL** [Platzer2010], **EDC** [Zhou et al 1994], ...
- **Advantages:** scalability
- **Difficulties:** invariant generation

Related work (Cont'd)

Deduction based approach [Platzer&Clarke 2008]

- **Basic idea:** extending **Floyd-Hoare-Naur inductive assertion method** to hybrid systems.
- **Elements:**
  - **A compositional modelling laguage**
  - **A Hoare logic-like specification logic**
  - **Invariant generation**
- **Well-known compositional modelling languages: hybrid programs** [Platzer&Clarke 2008], **HCSP** [He 1994, Zhou et al 1995], ...
- **Hybrid specification logics: DDL** [Platzer2008], **DADL** [Platzer2010], **EDC** [Zhou et al 1994], ...
- **Advantages:** scalability
- **Difficulties:** invariant generation

Related work (Cont'd)

Deduction based approach [Platzer&Clarke 2008]

- **Basic idea:** extending **Floyd-Hoare-Naur inductive assertion method** to hybrid systems.
- **Elements:**
  - **A compositional modelling laguage**
  - **A Hoare logic-like specification logic**
  - **Invariant generation**
- **Well-known compositional modelling languages: hybrid programs** [Platzer&Clarke 2008], **HCSP** [He 1994, Zhou et al 1995], ...
- **Hybrid specification logics: DDL** [Platzer2008], **DADL** [Platzer2010], **EDC** [Zhou et al 1994], ...
- **Advantages:** scalability
- **Difficulties:** invariant generation

Related work (Cont'd)

Deduction based approach [Platzer&Clarke 2008]

- **Basic idea:** extending **Floyd-Hoare-Naur inductive assertion method** to hybrid systems.
- **Elements:**
  - **A compositional modelling laguage**
  - **A Hoare logic-like specification logic**
  - **Invariant generation**
- **Well-known compositional modelling languages:** **hybrid programs** [Platzer&Clarke 2008], **HCSP** [He 1994, Zhou et al 1995], ...
- **Hybrid specification logics:** **DDL** [Platzer2008], **DADL** [Platzer2010], **EDC** [Zhou et al 1994], ...
- **Advantages:** scalability
- **Difficulties:** invariant generation

Related work (Cont'd)

Deduction based approach [Platzer&Clarke 2008]

- **Basic idea:** extending **Floyd-Hoare-Naur inductive assertion method** to hybrid systems.
- **Elements:**
  - **A compositional modelling laguage**
  - **A Hoare logic-like specification logic**
  - **Invariant generation**
- **Well-known compositional modelling languages: hybrid programs** [Platzer&Clarke 2008], **HCSP** [He 1994, Zhou et al 1995], ...
- **Hybrid specification logics: DDL** [Platzer2008], **DADL** [Platzer2010], **EDC** [Zhou et al 1994], ...
- **Advantages:** scalability
- **Difficulties:** invariant generation

Related work (Cont'd)

Deduction based approach [Platzer&Clarke 2008]

- **Basic idea:** extending **Floyd-Hoare-Naur inductive assertion method** to hybrid systems.
- **Elements:**
  - **A compositional modelling laguage**
  - **A Hoare logic-like specification logic**
  - **Invariant generation**
- **Well-known compositional modelling languages: hybrid programs** [Platzer&Clarke 2008], **HCSP** [He 1994, Zhou et al 1995], ...
- **Hybrid specification logics: DDL** [Platzer2008], **DADL** [Platzer2010], **EDC** [Zhou et al 1994], ...
- Advantages: scalability
- Difficulties: invariant generation

Related work (Cont'd)

Deduction based approach [Platzer&Clarke 2008]

- **Basic idea:** extending **Floyd-Hoare-Naur inductive assertion method** to hybrid systems.
- **Elements:**
  - **A compositional modelling laguage**
  - **A Hoare logic-like specification logic**
  - **Invariant generation**
- **Well-known compositional modelling languages: hybrid programs** [Platzer&Clarke 2008], **HCSP** [He 1994, Zhou et al 1995], ...
- **Hybrid specification logics: DDL** [Platzer2008], **DADL** [Platzer2010], **EDC** [Zhou et al 1994], ...
- **Advantages:** **scalability**
- **Difficulties:** **invariant generation**

Related work (Cont'd)

Deduction based approach [Platzer&Clarke 2008]

- **Basic idea:** extending **Floyd-Hoare-Naur inductive assertion method** to hybrid systems.
- **Elements:**
  - **A compositional modelling laguage**
  - **A Hoare logic-like specification logic**
  - **Invariant generation**
- **Well-known compositional modelling languages: hybrid programs** [Platzer&Clarke 2008], **HCSP** [He 1994, Zhou et al 1995], ...
- **Hybrid specification logics: DDL** [Platzer2008], **DADL** [Platzer2010], **EDC** [Zhou et al 1994], ...
- **Advantages:** scalability
- **Difficulties:** invariant generation

A grand challenge

How to design correct safety-critical hybrid-systems is a grand challenge in computer science and control theory

Our goal

to establish a systematic approach to formal design, analysis and verification of hybrid systems

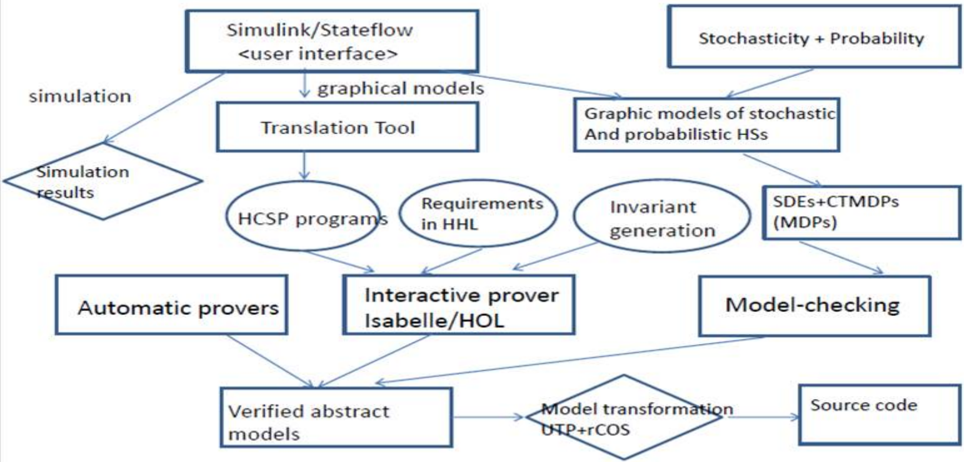
A grand challenge

How to design correct safety-critical hybrid-systems is a grand challenge in computer science and control theory

Our goal

to establish a systematic approach to formal design, analysis and verification of hybrid systems

Overview of Our Approach



Schedule and References

Schedule

- Lesson 1: Preliminaries + differential invariant generation
- Lesson 2: Controller synthesis
- Lesson 3: HCSP+HHL
- Lesson 4: HHL Prover + case study + demo

References

- N. Zhan, S. Wang and H. Zhao (2013): Formal Modelling, Analysis and Verification of Hybrid Systems. In the Theories of Programming, LNCS 8050.
- J. Liu, J. Lv, Z. Quan, N. Zhan, H. Zhao, C. Zhou and L. Zou (2010): A calculus for HCSP, in Proc. of APLAS 2010, LNCS 6461.
- J. Liu, N. Zhan and H. Zhao (2011): Computing semi-algebraic invariants for polynomial dynamical systems, in Proc. of EMSOFT'11.
- H. Zhao, N. Zhan, D. Kapur, and K.G. Larsen (2012): A "hybrid" approach for synthesizing optimal controllers of hybrid systems: A case study of the oil pump industrial example, in Proc. of FM 2012, LNCS 7436.
- H. Zhao, N. Zhan and D. Kapur (2013): Synthesizing switching controllers for hybrid systems by generating invariants, in Proc. of the Jifeng Festschrift, LNCS 8051.
- S. Wang, N. Zhan and D. Guelev (2012): An assume/guarantee based compositional calculus for HCSP, in Proc. of TAMC 2012, LNCS 7287.

Outline

Outline

- Let $\mathbb{K}$ be a number field, which can be either $\mathbb{Q}$ or $\mathbb{R}$.
- A ***monomial*** in n variables $x_1, x_2, \ldots, x_n$ (or briefly $\mathbf{x}$) is a product form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, or briefly $\mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n$. The number $\sum_{i=1}^{n} \alpha_i$ is called the ***degree*** of $\mathbf{x}^{\boldsymbol{\alpha}}$.
- A *polynomial* $p(\mathbf{x})$ in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is of the form $\sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where all $c_{\boldsymbol{\alpha}} \in \mathbb{K}$.
    - The ***degree*** $\deg(p)$ of p is the maximal degree of its component monomials.
    - A polynomial in $x_1, x_2, \ldots, x_n$ with degree d has at most $\binom{n+d}{d}$ many monomials.
    - The set of all polynomials in $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{K}$ form a *polynomial ring* $\mathbb{K}[\mathbf{x}]$.
- A ***parametric polynomial*** is of the form $\sum_{\boldsymbol{\alpha}} u_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where $u_{\boldsymbol{\alpha}} \in \mathbb{R}$ are not constants but undetermined parameters, can be regarded as a standard polynomial $p(\mathbf{u}, \mathbf{x})$ in $\mathbb{Q}[\mathbf{u}, \mathbf{x}]$.
    - A parametric polynomial with degree d (in $\mathbf{x}$) has at most $\binom{n+d}{d}$ many indeterminates.
    - For any $\mathbf{u}_0 \in \mathbb{R}^w$, $p_{\mathbf{u}_0}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ obtained by substituting $\mathbf{u}_0$ for $\mathbf{u}$ in $p(\mathbf{u}, \mathbf{x})$ is an *instantiation* of $p(\mathbf{u}, \mathbf{x})$.

- Let $\mathbb{K}$ be a number field, which can be either $\mathbb{Q}$ or $\mathbb{R}$.
- A ***monomial*** in n variables $x_1, x_2, \ldots, x_n$ (or briefly $\mathbf{x}$) is a product form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, or briefly $\mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n$. The number $\sum_{i=1}^{n} \alpha_i$ is called the ***degree*** of $\mathbf{x}^{\boldsymbol{\alpha}}$.
- A *polynomial* $p(\mathbf{x})$ in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is of the form $\sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where all $c_{\boldsymbol{\alpha}} \in \mathbb{K}$.
  - The ***degree*** $\deg(p)$ of p is the maximal degree of its component monomials.
  - A polynomial in $x_1, x_2, \ldots, x_n$ with degree d has at most $\binom{n+d}{d}$ many monomials.
  - The set of all polynomials in $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{K}$ form a *polynomial ring* $\mathbb{K}[\mathbf{x}]$.
- A ***parametric polynomial*** is of the form $\sum_{\boldsymbol{\alpha}} u_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where $u_{\boldsymbol{\alpha}} \in \mathbb{R}$ are not constants but undetermined parameters, can be regarded as a standard polynomial $p(\mathbf{u}, \mathbf{x})$ in $\mathbb{Q}[\mathbf{u}, \mathbf{x}]$.
  - A parametric polynomial with degree d (in $\mathbf{x}$) has at most $\binom{n+d}{d}$ many indeterminates.
  - For any $\mathbf{u}_0 \in \mathbb{R}^w$, $p_{\mathbf{u}_0}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ obtained by substituting $\mathbf{u}_0$ for $\mathbf{u}$ in $p(\mathbf{u}, \mathbf{x})$ is an *instantiation* of $p(\mathbf{u}, \mathbf{x})$.

- Let $\mathbb{K}$ be a number field, which can be either $\mathbb{Q}$ or $\mathbb{R}$.
- A ***monomial*** in n variables $x_1, x_2, \ldots, x_n$ (or briefly $\mathbf{x}$) is a product form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, or briefly $\mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n$. The number $\sum_{i=1}^{n} \alpha_i$ is called the ***degree*** of $\mathbf{x}^{\boldsymbol{\alpha}}$.
- A *polynomial* $p(\mathbf{x})$ in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is of the form $\sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where all $c_{\boldsymbol{\alpha}} \in \mathbb{K}$.
  - The ***degree*** $\deg(p)$ of p is the maximal degree of its component monomials.
  - A polynomial in $x_1, x_2, \ldots, x_n$ with degree d has at most $\binom{n+d}{d}$ many monomials.
  - The set of all polynomials in $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{K}$ form a *polynomial ring* $\mathbb{K}[\mathbf{x}]$.
- A ***parametric polynomial*** is of the form $\sum_{\boldsymbol{\alpha}} u_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where $u_{\boldsymbol{\alpha}} \in \mathbb{R}$ are not constants but undetermined parameters, can be regarded as a standard polynomial $p(\mathbf{u}, \mathbf{x})$ in $\mathbb{Q}[\mathbf{u}, \mathbf{x}]$.
  - A parametric polynomial with degree d (in $\mathbf{x}$) has at most $\binom{n+d}{d}$ many indeterminates.
  - For any $\mathbf{u}_0 \in \mathbb{R}^w$, $p_{\mathbf{u}_0}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ obtained by substituting $\mathbf{u}_0$ for $\mathbf{u}$ in $p(\mathbf{u}, \mathbf{x})$ is an *instantiation* of $p(\mathbf{u}, \mathbf{x})$.

- Let $\mathbb{K}$ be a number field, which can be either $\mathbb{Q}$ or $\mathbb{R}$.
- A ***monomial*** in n variables $x_1, x_2, \ldots, x_n$ (or briefly $\mathbf{x}$) is a product form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, or briefly $\mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n$. The number $\sum_{i=1}^{n} \alpha_i$ is called the ***degree*** of $\mathbf{x}^{\boldsymbol{\alpha}}$.
- A *polynomial* $p(\mathbf{x})$ in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is of the form $\sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where all $c_{\boldsymbol{\alpha}} \in \mathbb{K}$.
  - The ***degree*** $\deg(p)$ of p is the maximal degree of its component monomials.
  - A polynomial in $x_1, x_2, \ldots, x_n$ with degree d has at most $\binom{n+d}{d}$ many monomials.
  - The set of all polynomials in $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{K}$ form a *polynomial ring* $\mathbb{K}[\mathbf{x}]$.
- A ***parametric polynomial*** is of the form $\sum_{\boldsymbol{\alpha}} u_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where $u_{\boldsymbol{\alpha}} \in \mathbb{R}$ are not constants but undetermined parameters, can be regarded as a standard polynomial $p(\mathbf{u}, \mathbf{x})$ in $\mathbb{Q}[\mathbf{u}, \mathbf{x}]$.
  - A parametric polynomial with degree d (in $\mathbf{x}$) has at most $\binom{n+d}{d}$ many indeterminates.
  - For any $\mathbf{u}_0 \in \mathbb{R}^w$, $p_{\mathbf{u}_0}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ obtained by substituting $\mathbf{u}_0$ for $\mathbf{u}$ in $p(\mathbf{u}, \mathbf{x})$ is an *instantiation* of $p(\mathbf{u}, \mathbf{x})$.

- Let $\mathbb{K}$ be a number field, which can be either $\mathbb{Q}$ or $\mathbb{R}$.
- A ***monomial*** in n variables $x_1, x_2, \ldots, x_n$ (or briefly $\mathbf{x}$) is a product form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, or briefly $\mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n$. The number $\sum_{i=1}^{n} \alpha_i$ is called the ***degree*** of $\mathbf{x}^{\boldsymbol{\alpha}}$.
- A *polynomial* $p(\mathbf{x})$ in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is of the form $\sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where all $c_{\boldsymbol{\alpha}} \in \mathbb{K}$.
  - The ***degree*** $\deg(p)$ of p is the maximal degree of its component monomials.
  - A polynomial in $x_1, x_2, \ldots, x_n$ with degree d has at most $\binom{n+d}{d}$ many monomials.
  - The set of all polynomials in $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{K}$ form a *polynomial ring* $\mathbb{K}[\mathbf{x}]$.
- A ***parametric polynomial*** is of the form $\sum_{\boldsymbol{\alpha}} u_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where $u_{\boldsymbol{\alpha}} \in \mathbb{R}$ are not constants but undetermined parameters, can be regarded as a standard polynomial $p(\mathbf{u}, \mathbf{x})$ in $\mathbb{Q}[\mathbf{u}, \mathbf{x}]$.
  - A parametric polynomial with degree d (in $\mathbf{x}$) has at most $\binom{n+d}{d}$ many indeterminates.
  - For any $\mathbf{u}_0 \in \mathbb{R}^w$, $p_{\mathbf{u}_0}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ obtained by substituting $\mathbf{u}_0$ for $\mathbf{u}$ in $p(\mathbf{u}, \mathbf{x})$ is an *instantiation* of $p(\mathbf{u}, \mathbf{x})$.

- Let $\mathbb{K}$ be a number field, which can be either $\mathbb{Q}$ or $\mathbb{R}$.
- A ***monomial*** in n variables $x_1, x_2, \ldots, x_n$ (or briefly $\mathbf{x}$) is a product form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, or briefly $\mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n$. The number $\sum_{i=1}^{n} \alpha_i$ is called the ***degree*** of $\mathbf{x}^{\boldsymbol{\alpha}}$.
- A *polynomial* $p(\mathbf{x})$ in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is of the form $\sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where all $c_{\boldsymbol{\alpha}} \in \mathbb{K}$.
  - The ***degree*** $\deg(p)$ of p is the maximal degree of its component monomials.
  - A polynomial in $x_1, x_2, \ldots, x_n$ with degree d has at most $\binom{n+d}{d}$ many monomials.
  - The set of all polynomials in $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{K}$ form a *polynomial ring* $\mathbb{K}[\mathbf{x}]$.
- A ***parametric polynomial*** is of the form $\sum_{\boldsymbol{\alpha}} u_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where $u_{\boldsymbol{\alpha}} \in \mathbb{R}$ are not constants but undetermined parameters, can be regarded as a standard polynomial $p(\mathbf{u}, \mathbf{x})$ in $\mathbb{Q}[\mathbf{u}, \mathbf{x}]$.
  - A parametric polynomial with degree d (in $\mathbf{x}$) has at most $\binom{n+d}{d}$ many indeterminates.
  - For any $\mathbf{u}_0 \in \mathbb{R}^w$, $p_{\mathbf{u}_0}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ obtained by substituting $\mathbf{u}_0$ for $\mathbf{u}$ in $p(\mathbf{u}, \mathbf{x})$ is an *instantiation* of $p(\mathbf{u}, \mathbf{x})$.

- Let $\mathbb{K}$ be a number field, which can be either $\mathbb{Q}$ or $\mathbb{R}$.
- A ***monomial*** in n variables $x_1, x_2, \ldots, x_n$ (or briefly $\mathbf{x}$) is a product form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, or briefly $\mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n$. The number $\sum_{i=1}^n \alpha_i$ is called the ***degree*** of $\mathbf{x}^{\boldsymbol{\alpha}}$.
- A *polynomial* $p(\mathbf{x})$ in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is of the form $\sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where all $c_{\boldsymbol{\alpha}} \in \mathbb{K}$.
  - The ***degree*** $\deg(p)$ of p is the maximal degree of its component monomials.
  - A polynomial in $x_1, x_2, \ldots, x_n$ with degree d has at most $\binom{n+d}{d}$ many monomials.
  - The set of all polynomials in $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{K}$ form a *polynomial ring* $\mathbb{K}[\mathbf{x}]$.
- A ***parametric polynomial*** is of the form $\sum_{\boldsymbol{\alpha}} u_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where $u_{\boldsymbol{\alpha}} \in \mathbb{R}$ are not constants but undetermined parameters, can be regarded as a standard polynomial $p(\mathbf{u}, \mathbf{x})$ in $\mathbb{Q}[\mathbf{u}, \mathbf{x}]$.
  - A parametric polynomial with degree d (in $\mathbf{x}$) has at most $\binom{n+d}{d}$ many indeterminates.
  - For any $\mathbf{u}_0 \in \mathbb{R}^w$, $p_{\mathbf{u}_0}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ obtained by substituting $\mathbf{u}_0$ for $\mathbf{u}$ in $p(\mathbf{u}, \mathbf{x})$ is an *instantiation* of $p(\mathbf{u}, \mathbf{x})$.

- Let $\mathbb{K}$ be a number field, which can be either $\mathbb{Q}$ or $\mathbb{R}$.
- A ***monomial*** in n variables $x_1, x_2, \ldots, x_n$ (or briefly $\mathbf{x}$) is a product form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, or briefly $\mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n$. The number $\sum_{i=1}^{n} \alpha_i$ is called the ***degree*** of $\mathbf{x}^{\boldsymbol{\alpha}}$.
- A *polynomial* $p(\mathbf{x})$ in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is of the form $\sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where all $c_{\boldsymbol{\alpha}} \in \mathbb{K}$.
  - The ***degree*** $\deg(p)$ of p is the maximal degree of its component monomials.
  - A polynomial in $x_1, x_2, \ldots, x_n$ with degree d has at most $\binom{n+d}{d}$ many monomials.
  - The set of all polynomials in $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{K}$ form a *polynomial ring* $\mathbb{K}[\mathbf{x}]$.
- A ***parametric polynomial*** is of the form $\sum_{\boldsymbol{\alpha}} u_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where $u_{\boldsymbol{\alpha}} \in \mathbb{R}$ are not constants but undetermined parameters, can be regarded as a standard polynomial $p(\mathbf{u}, \mathbf{x})$ in $\mathbb{Q}[\mathbf{u}, \mathbf{x}]$.
  - A parametric polynomial with degree d (in $\mathbf{x}$) has at most $\binom{n+d}{d}$ many indeterminates.
  - For any $\mathbf{u}_0 \in \mathbb{R}^w$, $p_{\mathbf{u}_0}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ obtained by substituting $\mathbf{u}_0$ for $\mathbf{u}$ in $p(\mathbf{u}, \mathbf{x})$ is an *instantiation* of $p(\mathbf{u}, \mathbf{x})$.

- Let $\mathbb{K}$ be a number field, which can be either $\mathbb{Q}$ or $\mathbb{R}$.
- A ***monomial*** in n variables $x_1, x_2, \ldots, x_n$ (or briefly $\mathbf{x}$) is a product form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, or briefly $\mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n$. The number $\sum_{i=1}^{n} \alpha_i$ is called the ***degree*** of $\mathbf{x}^{\boldsymbol{\alpha}}$.
- A *polynomial* $p(\mathbf{x})$ in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is of the form $\sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where all $c_{\boldsymbol{\alpha}} \in \mathbb{K}$.
  - The ***degree*** $\deg(p)$ of p is the maximal degree of its component monomials.
  - A polynomial in $x_1, x_2, \ldots, x_n$ with degree d has at most $\binom{n+d}{d}$ many monomials.
  - The set of all polynomials in $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{K}$ form a *polynomial ring* $\mathbb{K}[\mathbf{x}]$.
- A ***parametric polynomial*** is of the form $\sum_{\boldsymbol{\alpha}} u_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where $u_{\boldsymbol{\alpha}} \in \mathbb{R}$ are not constants but undetermined parameters, can be regarded as a standard polynomial $p(\mathbf{u}, \mathbf{x})$ in $\mathbb{Q}[\mathbf{u}, \mathbf{x}]$.
  - A parametric polynomial with degree d (in $\mathbf{x}$) has at most $\binom{n+d}{d}$ many indeterminates.
  - For any $\mathbf{u}_0 \in \mathbb{R}^w$, $p_{\mathbf{u}_0}(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ obtained by substituting $\mathbf{u}_0$ for $\mathbf{u}$ in $p(\mathbf{u}, \mathbf{x})$ is an *instantiation* of $p(\mathbf{u}, \mathbf{x})$.

Polynomial ideal

Polynomial ideal

- A subset $I \subseteq \mathbb{K}[\mathbf{x}]$ is called an **ideal** if the following conditions are satisfied:
  1. $0 \in I$;
  2. If $p, g \in I$, then $p + g \in I$;
  3. If $p \in I$ and $h \in \mathbb{K}[\mathbf{x}]$, then $hp \in I$.
- Let $g_1, g_2, \ldots, g_s \in \mathbb{K}[\mathbf{x}]$, then $\langle g_1, g_2, \ldots, g_s \rangle \triangleq \{\sum_{i=1}^{s} h_i g_i : h_1, h_2, \ldots, h_s \in \mathbb{K}[\mathbf{x}]\}$ is an ideal *generated* by $g_1, g_2, \ldots, g_s$.
- If $I = \langle g_1, g_2, \ldots, g_s \rangle$, then $\{g_1, g_2, \ldots, g_s\}$ is called a ***basis*** of I.

Polynomial ideal

Polynomial ideal

- A subset $I \subseteq \mathbb{K}[\mathbf{x}]$ is called an **ideal** if the following conditions are satisfied:
  1. $0 \in I$;
  2. If $p, g \in I$, then $p + g \in I$;
  3. If $p \in I$ and $h \in \mathbb{K}[\mathbf{x}]$, then $hp \in I$.
- Let $g_1, g_2, \ldots, g_s \in \mathbb{K}[\mathbf{x}]$, then $\langle g_1, g_2, \ldots, g_s \rangle \widehat{=} \{\sum_{i=1}^{s} h_i g_i : h_1, h_2, \ldots, h_s \in \mathbb{K}[\mathbf{x}]\}$ is an ideal *generated* by $g_1, g_2, \ldots, g_s$.
- If $I = \langle g_1, g_2, \ldots, g_s \rangle$, then $\{g_1, g_2, \ldots, g_s\}$ is called a ***basis*** of I.

Polynomial ideal

Polynomial ideal

- A subset $I \subseteq \mathbb{K}[\mathbf{x}]$ is called an **ideal** if the following conditions are satisfied:
  1. $0 \in I$;
  2. If p, $g \in I$, then $p + g \in I$;
  3. If $p \in I$ and $h \in \mathbb{K}[\mathbf{x}]$, then $hp \in I$.
- Let $g_1, g_2, \ldots, g_s \in \mathbb{K}[\mathbf{x}]$, then $\langle g_1, g_2, \ldots, g_s \rangle \hat{=} \{\sum_{i=1}^{s} h_i g_i : h_1, h_2, \ldots, h_s \in \mathbb{K}[\mathbf{x}]\}$ is an ideal *generated* by $g_1, g_2, \ldots, g_s$.
- If $I = \langle g_1, g_2, \ldots, g_s \rangle$, then $\{g_1, g_2, \ldots, g_s\}$ is called a ***basis*** of I.

Hilbert Basis Theorem

Every ideal $I \subseteq \mathbb{K}[\mathbf{x}]$ has a finite basis, that is, $I = \langle g_1, g_2, \ldots, g_s \rangle$ for some $g_1, g_2, \ldots, g_s \in \mathbb{K}[\mathbf{x}]$.

Ascending Chain Theorem

For any ascending chain of ideals $I_1 \subseteq I_2 \subseteq \cdots \subseteq I_k \subseteq \cdots$ in $\mathbb{K}[\mathbf{x}]$, there exists an $N \in \mathbb{N}$ such that $I_k = I_N$ for any $k \geq N$.

Gröbner basis

Definitions

- **lexicographic ording:** Suppose $x_1 \succ x_2 \succ \cdots \succ x_n$, then the ***lexicographic* (lex) order** $\succ$ is a total ordering on the set of monomials $\mathbf{x}^{\boldsymbol{\alpha}}$ defined by: $\mathbf{x}^{\boldsymbol{\alpha}} \succ \mathbf{x}^{\boldsymbol{\beta}}$ iff there exists $1 \leq i \leq n$ such that $\alpha_i > \beta_i$, and $\alpha_j = \beta_j$ for all $1 \leq j < i$.
  - The lex is ***well-ordering***.
  - The lex is preserved by multiplication.
- Given a polynomial $g \in \mathbb{K}[\mathbf{x}]$, rearrange the monomials in p by $\succ$ in a descending order as $g = c_1\mathbf{x}^{\boldsymbol{\alpha}_1} + c_2\mathbf{x}^{\boldsymbol{\alpha}_2} + \cdots + c_k\mathbf{x}^{\boldsymbol{\alpha}_k}$, where all c_i's are nonzero. Then
  - **the *leading term*** $\mathrm{lt}(g)$ of g is $c_1\mathbf{x}^{\boldsymbol{\alpha}_1}$;
  - **the *leading coefficient*** $\mathrm{lc}(g)$ of g is c_1;
  - **the *leading monomial*** $\mathrm{lm}(g)$ of g is $\mathbf{x}^{\boldsymbol{\alpha}_1}$.

Gröbner basis

Definitions

- **lexicographic ording:** Suppose $x_1 \succ x_2 \succ \cdots \succ x_n$, then the ***lexicographic* (lex) order** $\succ$ is a total ordering on the set of monomials $\mathbf{x}^{\boldsymbol{\alpha}}$ defined by: $\mathbf{x}^{\boldsymbol{\alpha}} \succ \mathbf{x}^{\boldsymbol{\beta}}$ iff there exists $1 \leq i \leq n$ such that $\alpha_i > \beta_i$, and $\alpha_j = \beta_j$ for all $1 \leq j < i$.
  - The lex is ***well-ordering***.
  - The lex is preserved by multiplication.
- Given a polynomial $g \in \mathbb{K}[\mathbf{x}]$, rearrange the monomials in p by $\succ$ in a descending order as $g = c_1\mathbf{x}^{\boldsymbol{\alpha_1}} + c_2\mathbf{x}^{\boldsymbol{\alpha_2}} + \cdots + c_k\mathbf{x}^{\boldsymbol{\alpha_k}}$, where all c_i's are nonzero. Then
  - **the *leading term*** $\mathrm{lt}(g)$ of g is $c_1\mathbf{x}^{\boldsymbol{\alpha_1}}$;
  - **the *leading coefficient*** $\mathrm{lc}(g)$ of g is c_1;
  - **the *leading monomial*** $\mathrm{lm}(g)$ of g is $\mathbf{x}^{\boldsymbol{\alpha_1}}$.

Gröbner basis

Definitions

- **lexicographic ording:** Suppose $x_1 \succ x_2 \succ \cdots \succ x_n$, then the ***lexicographic* (lex) order** $\succ$ is a total ordering on the set of monomials $\mathbf{x}^{\alpha}$ defined by: $\mathbf{x}^{\alpha} \succ \mathbf{x}^{\beta}$ iff there exists $1 \le i \le n$ such that $\alpha_i > \beta_i$, and $\alpha_j = \beta_j$ for all $1 \le j < i$.
  - The lex is ***well-ordering***.
  - The lex is preserved by multiplication.
- Given a polynomial $g \in \mathbb{K}[\mathbf{x}]$, rearrange the monomials in p by $\succ$ in a descending order as $g = c_1\mathbf{x}^{\alpha_1} + c_2\mathbf{x}^{\alpha_2} + \cdots + c_k\mathbf{x}^{\alpha_k}$, where all c_i's are nonzero. Then
  - **the *leading term*** $\mathrm{lt}(g)$ of g is $c_1\mathbf{x}^{\alpha_1}$;
  - **the *leading coefficient*** $\mathrm{lc}(g)$ of g is c_1;
  - **the *leading monomial*** $\mathrm{lm}(g)$ of g is $\mathbf{x}^{\alpha_1}$.

Gröbner basis

Definitions

- **lexicographic ording:** Suppose $x_1 \succ x_2 \succ \cdots \succ x_n$, then the ***lexicographic* (lex) order** $\succ$ is a total ordering on the set of monomials $\mathbf{x}^{\boldsymbol{\alpha}}$ defined by: $\mathbf{x}^{\boldsymbol{\alpha}} \succ \mathbf{x}^{\boldsymbol{\beta}}$ iff there exists $1 \leq i \leq n$ such that $\alpha_i > \beta_i$, and $\alpha_j = \beta_j$ for all $1 \leq j < i$.
  - The lex is ***well-ordering***.
  - The lex is preserved by multiplication.
- Given a polynomial $g \in \mathbb{K}[\mathbf{x}]$, rearrange the monomials in p by $\succ$ in a descending order as $g = c_1\mathbf{x}^{\boldsymbol{\alpha_1}} + c_2\mathbf{x}^{\boldsymbol{\alpha_2}} + \cdots + c_k\mathbf{x}^{\boldsymbol{\alpha_k}}$, where all c_i's are nonzero. Then
  - **the *leading term*** $\mathrm{lt}(g)$ of g is $c_1\mathbf{x}^{\boldsymbol{\alpha_1}}$;
  - **the *leading coefficient*** $\mathrm{lc}(g)$ of g is c_1;
  - **the *leading monomial*** $\mathrm{lm}(g)$ of g is $\mathbf{x}^{\boldsymbol{\alpha_1}}$.

Gröbner basis

Definitions

- **lexicographic ording:** Suppose $x_1 \succ x_2 \succ \cdots \succ x_n$, then the ***lexicographic* (lex) order** $\succ$ is a total ordering on the set of monomials $\mathbf{x}^{\boldsymbol{\alpha}}$ defined by: $\mathbf{x}^{\boldsymbol{\alpha}} \succ \mathbf{x}^{\boldsymbol{\beta}}$ iff there exists $1 \leq i \leq n$ such that $\alpha_i > \beta_i$, and $\alpha_j = \beta_j$ for all $1 \leq j < i$.
  - The lex is ***well-ordering***.
  - The lex is preserved by multiplication.
- Given a polynomial $g \in \mathbb{K}[\mathbf{x}]$, rearrange the monomials in p by $\succ$ in a descending order as $g = c_1\mathbf{x}^{\boldsymbol{\alpha_1}} + c_2\mathbf{x}^{\boldsymbol{\alpha_2}} + \cdots + c_k\mathbf{x}^{\boldsymbol{\alpha_k}}$, where all c_i's are nonzero. Then
  - **the *leading term*** $\mathrm{lt}(g)$ of g is $c_1\mathbf{x}^{\boldsymbol{\alpha_1}}$;
  - **the *leading coefficient*** $\mathrm{lc}(g)$ of g is c_1;
  - **the *leading monomial*** $\mathrm{lm}(g)$ of g is $\mathbf{x}^{\boldsymbol{\alpha_1}}$.

Gröbner basis

Definitions

- **lexicographic ording:** Suppose $x_1 \succ x_2 \succ \cdots \succ x_n$, then the ***lexicographic* (lex) order** $\succ$ is a total ordering on the set of monomials $\mathbf{x}^{\boldsymbol{\alpha}}$ defined by: $\mathbf{x}^{\boldsymbol{\alpha}} \succ \mathbf{x}^{\boldsymbol{\beta}}$ iff there exists $1 \leq i \leq n$ such that $\alpha_i > \beta_i$, and $\alpha_j = \beta_j$ for all $1 \leq j < i$.
  - The lex is ***well-ordering***.
  - The lex is preserved by multiplication.
- Given a polynomial $g \in \mathbb{K}[\mathbf{x}]$, rearrange the monomials in p by $\succ$ in a descending order as $g = c_1\mathbf{x}^{\boldsymbol{\alpha_1}} + c_2\mathbf{x}^{\boldsymbol{\alpha_2}} + \cdots + c_k\mathbf{x}^{\boldsymbol{\alpha_k}}$, where all c_i's are nonzero. Then
  - **the *leading term*** $\mathrm{lt}(g)$ of g is $c_1\mathbf{x}^{\boldsymbol{\alpha_1}}$;
  - **the *leading coefficient*** $\mathrm{lc}(g)$ of g is c_1;
  - **the *leading monomial*** $\mathrm{lm}(g)$ of g is $\mathbf{x}^{\boldsymbol{\alpha_1}}$.

Gröbner basis

Definitions

- **lexicographic ording:** Suppose $x_1 \succ x_2 \succ \cdots \succ x_n$, then the ***lexicographic* (lex) order** $\succ$ is a total ordering on the set of monomials $\mathbf{x}^{\boldsymbol{\alpha}}$ defined by: $\mathbf{x}^{\boldsymbol{\alpha}} \succ \mathbf{x}^{\boldsymbol{\beta}}$ iff there exists $1 \leq i \leq n$ such that $\alpha_i > \beta_i$, and $\alpha_j = \beta_j$ for all $1 \leq j < i$.
  - The lex is ***well-ordering***.
  - The lex is preserved by multiplication.
- Given a polynomial $g \in \mathbb{K}[\mathbf{x}]$, rearrange the monomials in p by $\succ$ in a descending order as $g = c_1\mathbf{x}^{\boldsymbol{\alpha_1}} + c_2\mathbf{x}^{\boldsymbol{\alpha_2}} + \cdots + c_k\mathbf{x}^{\boldsymbol{\alpha_k}}$, where all c_i's are nonzero. Then
  - **the *leading term*** $\mathrm{lt}(g)$ of g is $c_1\mathbf{x}^{\boldsymbol{\alpha_1}}$;
  - **the *leading coefficient*** $\mathrm{lc}(g)$ of g is c_1;
  - **the *leading monomial*** $\mathrm{lm}(g)$ of g is $\mathbf{x}^{\boldsymbol{\alpha_1}}$.

Gröbner Basis (Cont'd)

Reduction

- **One step reduction:** For a polynomial $p \in \mathbb{K}[\mathbf{x}]$, if p has a nonzero term $c_\beta \mathbf{x}^\beta$ and $\mathbf{x}^\beta = \mathbf{x}^\gamma \mathrm{lm}(g)$ for some $\gamma \in \mathbb{N}^n$, then we say p is ***reducible* modulo** g, and call

$$p' = p - \frac{c_\beta}{\mathrm{lc}(g)} \mathbf{x}^\gamma g$$

the one-step *reduction* of p **modulo** g.

- Given a finite set of polynomials $G \subsetneq \mathbb{K}[\mathbf{x}]$ and a polynomial $p \in \mathbb{K}[\mathbf{x}]$, we can do a muli-step reduction on p using polynomials in G, until p is reduced to p^* which is not further reducible modulo G.
- p^* is called the *normal form* of p w.r.t. G ($\mathrm{nf}(p, G)$). Note that
  - the above process of reduction is guaranteed to terminate;
  - however, the final result $\mathrm{nf}(p, G)$ may vary, depending on the sequence of polynomials chosen from G during reduction.

Gröbner Basis (Cont'd)

Reduction

- **One step reduction:** For a polynomial $p \in \mathbb{K}[\mathbf{x}]$, if p has a nonzero term $c_\beta \mathbf{x}^\beta$ and $\mathbf{x}^\beta = \mathbf{x}^\gamma \mathrm{lm}(g)$ for some $\gamma \in \mathbb{N}^n$, then we say p is ***reducible* modulo** g, and call

$$p' = p - \frac{c_\beta}{\mathrm{lc}(g)} \mathbf{x}^\gamma g$$

the one-step *reduction* of p **modulo** g.

- Given a finite set of polynomials $G \subsetneq \mathbb{K}[\mathbf{x}]$ and a polynomial $p \in \mathbb{K}[\mathbf{x}]$, we can do a muli-step reduction on p using polynomials in G, until p is reduced to p^* which is not further reducible modulo G.
- p^* is called the *normal form* of p w.r.t. G ($\mathrm{nf}(p, G)$). Note that
  - the above process of reduction is guaranteed to terminate;
  - however, the final result $\mathrm{nf}(p, G)$ may vary, depending on the sequence of polynomials chosen from G during reduction.

Gröbner Basis (Cont'd)

Reduction

- **One step reduction:** For a polynomial $p \in \mathbb{K}[\mathbf{x}]$, if p has a nonzero term $c_\beta \mathbf{x}^\beta$ and $\mathbf{x}^\beta = \mathbf{x}^\gamma \mathrm{lm}(g)$ for some $\gamma \in \mathbb{N}^n$, then we say p is ***reducible* modulo** g, and call

$$p' = p - \frac{c_\beta}{\mathrm{lc}(g)} \mathbf{x}^\gamma g$$

  the one-step *reduction* of p **modulo** g.
- Given a finite set of polynomials $G \subsetneq \mathbb{K}[\mathbf{x}]$ and a polynomial $p \in \mathbb{K}[\mathbf{x}]$, we can do a muli-step reduction on p using polynomials in G, until p is reduced to p^* which is not further reducible modulo G.
- p^* is called **the *normal form*** of p w.r.t. G ($\mathrm{nf}(p, G)$). Note that
  - the above process of reduction is guaranteed to terminate;
  - however, the final result $\mathrm{nf}(p, G)$ may vary, depending on the sequence of polynomials chosen from G during reduction.

Gröbner Basis (Cont'd)

Reduction

- **One step reduction:** For a polynomial $p \in \mathbb{K}[\mathbf{x}]$, if p has a nonzero term $c_\beta \mathbf{x}^\beta$ and $\mathbf{x}^\beta = \mathbf{x}^\gamma \mathrm{lm}(g)$ for some $\gamma \in \mathbb{N}^n$, then we say p is ***reducible* modulo** g, and call

$$p' = p - \frac{c_\beta}{\mathrm{lc}(g)} \mathbf{x}^\gamma g$$

  the one-step *reduction* of p **modulo** g.
- Given a finite set of polynomials $G \subsetneq \mathbb{K}[\mathbf{x}]$ and a polynomial $p \in \mathbb{K}[\mathbf{x}]$, we can do a muli-step reduction on p using polynomials in G, until p is reduced to p^* which is not further reducible modulo G.
- p^* is called **the *normal form*** of p w.r.t. G ($\mathrm{nf}(p, G)$). Note that
  - the above process of reduction is guaranteed to terminate;
  - however, the final result $\mathrm{nf}(p, G)$ may vary, depending on the sequence of polynomials chosen from G during reduction.

Gröbner Basis (Cont'd)

Reduction

- **One step reduction:** For a polynomial $p \in \mathbb{K}[\mathbf{x}]$, if p has a nonzero term $c_\beta \mathbf{x}^\beta$ and $\mathbf{x}^\beta = \mathbf{x}^\gamma \mathrm{lm}(g)$ for some $\gamma \in \mathbb{N}^n$, then we say p is ***reducible* modulo** g, and call

$$p' = p - \frac{c_\beta}{\mathrm{lc}(g)} \mathbf{x}^\gamma g$$

  the one-step *reduction* of p **modulo** g.
- Given a finite set of polynomials $G \subsetneq \mathbb{K}[\mathbf{x}]$ and a polynomial $p \in \mathbb{K}[\mathbf{x}]$, we can do a muli-step reduction on p using polynomials in G, until p is reduced to p^* which is not further reducible modulo G.
- p^* is called **the *normal form*** of p w.r.t. G ($\mathrm{nf}(p, G)$). Note that
  - the above process of reduction is guaranteed to terminate;
  - however, the final result $\mathrm{nf}(p, G)$ may vary, depending on the sequence of polynomials chosen from G during reduction.

Gröbner Basis (Cont'd)

Gröbner basis

- Given a monomial ordering, then every ideal $I \subseteq \mathbb{K}[\mathbf{x}]$ other than $\{0\}$ has a basis $G = \{g_1, g_2, \ldots, g_s\}$, such that for any $p \in \mathbb{K}[\mathbf{x}]$, $\mathrm{nf}(p, G)$ is unique. Such G is called a **Gröbner basis** of I.
- Let G be a Gröbner basis of an ideal $I \subseteq \mathbb{K}[\mathbf{x}]$. Then for any $p \in \mathbb{K}[\mathbf{x}]$, $p \in I$ iff $\mathrm{nf}(p, G) = 0$.
- For any ideal $I = \langle h_1, h_2, \ldots, h_m \rangle \subseteq \mathbb{K}[\mathbf{x}]$, the Gröbner basis G of I can be computed from the h_i's using ***Buchberger's Algorithm***.

Gröbner Basis (Cont'd)

Gröbner basis

- Given a monomial ordering, then every ideal $I \subseteq \mathbb{K}[\mathbf{x}]$ other than $\{0\}$ has a basis $G = \{g_1, g_2, \ldots, g_s\}$, such that for any $p \in \mathbb{K}[\mathbf{x}]$, $\mathrm{nf}(p, G)$ is unique. Such G is called a **Gröbner basis** of I.
- Let G be a Gröbner basis of an ideal $I \subseteq \mathbb{K}[\mathbf{x}]$. Then for any $p \in \mathbb{K}[\mathbf{x}]$, $p \in I$ iff $\mathrm{nf}(p, G) = 0$.
- For any ideal $I = \langle h_1, h_2, \ldots, h_m \rangle \subseteq \mathbb{K}[\mathbf{x}]$, the Gröbner basis G of I can be computed from the h_i's using ***Buchberger's Algorithm***.

Gröbner Basis (Cont'd)

Gröbner basis

- Given a monomial ordering, then every ideal $I \subseteq \mathbb{K}[\mathbf{x}]$ other than $\{0\}$ has a basis $G = \{g_1, g_2, \ldots, g_s\}$, such that for any $p \in \mathbb{K}[\mathbf{x}]$, $\mathrm{nf}(p, G)$ is unique. Such G is called a **Gröbner basis** of I.
- Let G be a Gröbner basis of an ideal $I \subseteq \mathbb{K}[\mathbf{x}]$. Then for any $p \in \mathbb{K}[\mathbf{x}]$, $p \in I$ iff $\mathrm{nf}(p, G) = 0$.
- For any ideal $I = \langle h_1, h_2, \ldots, h_m \rangle \subseteq \mathbb{K}[\mathbf{x}]$, the Gröbner basis G of I can be computed from the h_i's using ***Buchberger's Algorithm***.

Buchberger's Algorithm for Computing GB

- S-polynomial: If $\mathrm{lm}(f) = \mathbf{x}^{\boldsymbol{\alpha}}$, $\mathrm{lm}(g) = \mathbf{x}^{\boldsymbol{\beta}}$, then let $\gamma_i \mathrel{\widehat{=}} \max(\alpha_i, \beta_i)$,

$$S(f,g) \mathrel{\widehat{=}} \frac{\mathbf{x}^{\boldsymbol{\gamma}}}{\mathrm{lt}(f)} \cdot f - \frac{\mathbf{x}^{\boldsymbol{\gamma}}}{\mathrm{lt}(g)} \cdot g$$

- Buchberger's Algorithm

Input: $F = \{f_1, f_2, \ldots, f_m\}$
Output: a Gröbner basis $G = \{g_1, g_2, \ldots, g_s\}$ of $I \mathrel{\widehat{=}} \langle f_1, f_2, \ldots, f_m \rangle$

$G := F$;
repeat
  $G' := G$;
  for *all* $p, q \in G', p \neq q$ **do**
    $s := S(p, q)$; $r := \mathrm{nf}(s, G')$;
    if $r \neq 0$ **then**
      $G := G \cup \{r\}$;
until $G = G'$;

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1$, $p_2 \mathrel{\widehat{=}} xy^2 - x$, $I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1$, $p_2 \mathrel{\widehat{=}} xy^2 - x$, $I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ p_2 \mathrel{\widehat{=}} xy^2 - x,\ I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \widehat{=} x^2y - 1$, $p_2 \widehat{=} xy^2 - x$, $I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \widehat{=} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \widehat{=} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ p_2 \mathrel{\widehat{=}} xy^2 - x,\ I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ \ p_2 \mathrel{\widehat{=}} xy^2 - x,\ \ I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ \ p_2 \mathrel{\widehat{=}} xy^2 - x,\ \ I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ \ p_2 \mathrel{\widehat{=}} xy^2 - x,\ \ I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ \ p_2 \mathrel{\widehat{=}} xy^2 - x,\ \ I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ p_2 \mathrel{\widehat{=}} xy^2 - x,\ I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ p_2 \mathrel{\widehat{=}} xy^2 - x,\ I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ p_2 \mathrel{\widehat{=}} xy^2 - x,\ I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1$, $p_2 \mathrel{\widehat{=}} xy^2 - x$, $I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1$, $p_2 \mathrel{\widehat{=}} xy^2 - x$, $I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1$, $p_2 \mathrel{\widehat{=}} xy^2 - x$, $I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1$, $p_2 \mathrel{\widehat{=}} xy^2 - x$, $I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Buchberger's Algorithm: An Example

- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ \ p_2 \mathrel{\widehat{=}} xy^2 - x,\ \ I = \langle p_1, p_2 \rangle$. Use lex order $x \succ y$.
- $G = \{p_1, p_2\}$
  - $S(p_1, p_2) = y \cdot p_1 - x \cdot p_2 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = x^2 - y \neq 0$
- $G = \{p_1, p_2, p_3 \mathrel{\widehat{=}} x^2 - y\}$
  - $S(p_1, p_3) = p_1 - y \cdot p_3 = y^2 - 1$
  - $\mathrm{nf}(y^2 - 1, G) = y^2 - 1 \neq 0$
- $G = \{p_1, p_2, p_3, p_4 \mathrel{\widehat{=}} y^2 - 1\}$
  - $S(p_1, p_4) = y \cdot p_1 - x^2 \cdot p_4 = x^2 - y$
  - $\mathrm{nf}(x^2 - y, G) = 0 \quad (\because x^2 - y \xrightarrow{p_3} 0)$
  - $S(p_2, p_3) = x \cdot p_2 - y^2 \cdot p_3 = -x^2 + y^3$
  - $\mathrm{nf}(-x^2 + y^3, G) = 0 \quad (\because -x^2 + y^3 \xrightarrow{p_3} y^3 - y \xrightarrow{p_4} 0)$
  - $S(p_2, p_4) = p_2 - x \cdot p_4 = 0$
  - $\mathrm{nf}(0, G) = 0$
  - $S(p_3, p_4) = y^2 \cdot p_3 - x^2 \cdot p_4 = x^2 - y^3$
  - $\mathrm{nf}(x^2 - y^3, G) = 0 \quad (\because x^2 - y^3 \xrightarrow{p_3} -y^3 + y \xrightarrow{p_4} 0)$
- $\therefore G = \{p_1, p_2, p_3, p_4\}$

Application of Gröbner Basis: PIMP

- PIMP: Polynomial Ideal Membership Problem
- $p_1 \,\widehat{=}\, x^2y - 1,\ p_2 \,\widehat{=}\, xy^2 - x,\ I = \langle p_1, p_2 \rangle$.
- $G = \{p_1, p_2, p_3, p_4\}$
  $= \{x^2y - 1, xy^2 - x, x^2 - y, y^2 - 1\}$
- $h_1 \,\widehat{=}\, x^2y + x - 2y^2$

$$h_1 \xrightarrow[x^2y]{p_1} x - 2y^2 + 1 \xrightarrow[-2y^2]{p_4} x - 1 \not\rightarrow$$

  so $\mathrm{nf}(h_1, G) = x - 1$, $h_1 \notin I$
- $h_2 \,\widehat{=}\, x^3 + xy^2 - xy - x$

$$h_1 \xrightarrow[x^3]{p_3} xy^2 - x \xrightarrow[xy^2]{p_2} 0$$

  so $\mathrm{nf}(h_2, G) = 0$, $h_2 \in I$

Application of Gröbner Basis: PIMP

- PIMP: Polynomial Ideal Membership Problem
- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ \ p_2 \mathrel{\widehat{=}} xy^2 - x,\ \ I = \langle p_1, p_2 \rangle$.
- $G = \{p_1, p_2, p_3, p_4\}$
  $= \{x^2y - 1, xy^2 - x, x^2 - y, y^2 - 1\}$
- $h_1 \mathrel{\widehat{=}} x^2y + x - 2y^2$

$$h_1 \xrightarrow[x^2y]{p_1} x - 2y^2 + 1 \xrightarrow[-2y^2]{p_4} x - 1 \not\rightarrow$$

  so $\mathrm{nf}(h_1, G) = x - 1,\ h_1 \notin I$
- $h_2 \mathrel{\widehat{=}} x^3 + xy^2 - xy - x$

$$h_1 \xrightarrow[x^3]{p_3} xy^2 - x \xrightarrow[xy^2]{p_2} 0$$

  so $\mathrm{nf}(h_2, G) = 0,\ h_2 \in I$

Application of Gröbner Basis: PIMP

- PIMP: Polynomial Ideal Membership Problem
- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ p_2 \mathrel{\widehat{=}} xy^2 - x,\ I = \langle p_1, p_2 \rangle$.
- $G = \{p_1, p_2, p_3, p_4\}$
  $= \{x^2y - 1, xy^2 - x, x^2 - y, y^2 - 1\}$
- $h_1 \mathrel{\widehat{=}} x^2y + x - 2y^2$

$$h_1 \xrightarrow[x^2y]{p_1} x - 2y^2 + 1 \xrightarrow[-2y^2]{p_4} x - 1 \not\rightarrow$$

  so $\mathrm{nf}(h_1, G) = x - 1,\ h_1 \notin I$
- $h_2 \mathrel{\widehat{=}} x^3 + xy^2 - xy - x$

$$h_1 \xrightarrow[x^3]{p_3} xy^2 - x \xrightarrow[xy^2]{p_2} 0$$

  so $\mathrm{nf}(h_2, G) = 0,\ h_2 \in I$

Application of Gröbner Basis: PIMP

- PIMP: Polynomial Ideal Membership Problem
- $p_1 \widehat{=} x^2y - 1,\ p_2 \widehat{=} xy^2 - x,\ I = \langle p_1, p_2 \rangle$.
- $G = \{p_1, p_2, p_3, p_4\}$
  $= \{x^2y - 1, xy^2 - x, x^2 - y, y^2 - 1\}$
- $h_1 \widehat{=} x^2y + x - 2y^2$

$$h_1 \xrightarrow[x^2y]{p_1} x - 2y^2 + 1 \xrightarrow[-2y^2]{p_4} x - 1 \not\rightarrow$$

  so $\mathrm{nf}(h_1, G) = x - 1$, $h_1 \notin I$
- $h_2 \widehat{=} x^3 + xy^2 - xy - x$

$$h_1 \xrightarrow[x^3]{p_3} xy^2 - x \xrightarrow[xy^2]{p_2} 0$$

  so $\mathrm{nf}(h_2, G) = 0$, $h_2 \in I$

Application of Gröbner Basis: PIMP

- PIMP: Polynomial Ideal Membership Problem
- $p_1 \mathrel{\widehat{=}} x^2y - 1,\ p_2 \mathrel{\widehat{=}} xy^2 - x,\ I = \langle p_1, p_2 \rangle.$
- $G = \{p_1, p_2, p_3, p_4\}$
  $= \{x^2y - 1, xy^2 - x, x^2 - y, y^2 - 1\}$
- $h_1 \mathrel{\widehat{=}} x^2y + x - 2y^2$

$$h_1 \xrightarrow[x^2y]{p_1} x - 2y^2 + 1 \xrightarrow[-2y^2]{p_4} x - 1 \not\rightarrow$$

  so $\mathrm{nf}(h_1, G) = x - 1$, $h_1 \notin I$
- $h_2 \mathrel{\widehat{=}} x^3 + xy^2 - xy - x$

$$h_1 \xrightarrow[x^3]{p_3} xy^2 - x \xrightarrow[xy^2]{p_2} 0$$

  so $\mathrm{nf}(h_2, G) = 0$, $h_2 \in I$

Outline

First-order Theory $T(\mathbb{R})$ of Reals

Syntax

- The language of $T(\mathbb{R})$ consists of
  - variables: $x, y, z, \ldots, x_1, x_2, \ldots$, which are interpreted over $\mathbb{R}$;
  - relational symbols: $>, <, \geq, \leq, =, \neq$;
  - Boolean connectives: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \ldots$; and
  - quantifiers: $\forall, \exists$.
- A *term* of $T(\mathbb{R})$ over a finite set of variables $\{x_1, x_2, \ldots, x_n\}$ is a polynomial $p \in \mathbb{Q}[x_1, x_2, \ldots, x_n]$.
- An *atomic formula* of $T(\mathbb{R})$ is of the form $p \rhd 0$, where $\rhd$ is any relational symbol.
- A *quantifier-free formula* (QFF) of $T(\mathbb{R})$ is a Boolean combination of atomic formulas.
- A generic formula of $T(\mathbb{R})$ is built up from atomic formulas using Boolean connectives as well as quantifiers.

First-order Theory $T(\mathbb{R})$ of Reals

Syntax

- The language of $T(\mathbb{R})$ consists of
  - variables: $x, y, z, \ldots, x_1, x_2, \ldots$, which are interpreted over $\mathbb{R}$;
  - relational symbols: $>, <, \geq, \leq, =, \neq$;
  - Boolean connectives: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \ldots$; and
  - quantifiers: $\forall, \exists$.
- A *term* of $T(\mathbb{R})$ over a finite set of variables $\{x_1, x_2, \ldots, x_n\}$ is a polynomial $p \in \mathbb{Q}[x_1, x_2, \ldots, x_n]$.
- An *atomic formula* of $T(\mathbb{R})$ is of the form $p \rhd 0$, where $\rhd$ is any relational symbol.
- A *quantifier-free formula* (QFF) of $T(\mathbb{R})$ is a Boolean combination of atomic formulas.
- A generic formula of $T(\mathbb{R})$ is built up from atomic formulas using Boolean connectives as well as quantifiers.

First-order Theory $T(\mathbb{R})$ of Reals

Syntax

- The language of $T(\mathbb{R})$ consists of
  - variables: $x, y, z, \ldots, x_1, x_2, \ldots$, which are interpreted over $\mathbb{R}$;
  - relational symbols: $>, <, \geq, \leq, =, \neq$;
  - Boolean connectives: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \ldots$; and
  - quantifiers: $\forall, \exists$.
- A *term* of $T(\mathbb{R})$ over a finite set of variables $\{x_1, x_2, \ldots, x_n\}$ is a polynomial $p \in \mathbb{Q}[x_1, x_2, \ldots, x_n]$.
- An *atomic formula* of $T(\mathbb{R})$ is of the form $p \triangleright 0$, where $\triangleright$ is any relational symbol.
- A *quantifier-free formula* (QFF) of $T(\mathbb{R})$ is a Boolean combination of atomic formulas.
- A generic formula of $T(\mathbb{R})$ is built up from atomic formulas using Boolean connectives as well as quantifiers.

First-order Theory $T(\mathbb{R})$ of Reals

Syntax

- The language of $T(\mathbb{R})$ consists of
  - variables: $x, y, z, \ldots, x_1, x_2, \ldots$, which are interpreted over $\mathbb{R}$;
  - relational symbols: $>, <, \geq, \leq, =, \neq$;
  - Boolean connectives: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \ldots$; and
  - quantifiers: $\forall, \exists$.
- A *term* of $T(\mathbb{R})$ over a finite set of variables $\{x_1, x_2, \ldots, x_n\}$ is a polynomial $p \in \mathbb{Q}[x_1, x_2, \ldots, x_n]$.
- An *atomic formula* of $T(\mathbb{R})$ is of the form $p \rhd 0$, where $\rhd$ is any relational symbol.
- A *quantifier-free formula* (QFF) of $T(\mathbb{R})$ is a Boolean combination of atomic formulas.
- A generic formula of $T(\mathbb{R})$ is built up from atomic formulas using Boolean connectives as well as quantifiers.

First-order Theory $T(\mathbb{R})$ of Reals

Syntax

- The language of $T(\mathbb{R})$ consists of
  - variables: $x, y, z, \ldots, x_1, x_2, \ldots$, which are interpreted over $\mathbb{R}$;
  - relational symbols: $>, <, \geq, \leq, =, \neq$;
  - Boolean connectives: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \ldots$; and
  - quantifiers: $\forall, \exists$.
- A *term* of $T(\mathbb{R})$ over a finite set of variables $\{x_1, x_2, \ldots, x_n\}$ is a polynomial $p \in \mathbb{Q}[x_1, x_2, \ldots, x_n]$.
- An *atomic formula* of $T(\mathbb{R})$ is of the form $p \triangleright 0$, where $\triangleright$ is any relational symbol.
- A *quantifier-free formula* (QFF) of $T(\mathbb{R})$ is a Boolean combination of atomic formulas.
- A generic formula of $T(\mathbb{R})$ is built up from atomic formulas using Boolean connectives as well as quantifiers.

Quantifier Elimination

Quantifier Elimination Property

- A theory $\mathcal{T}$ is said to have **quantifier elimination property**, if for any formula φ in $\mathcal{T}$, there exists a quantifier-free formula φ_{QF} which only contains *free* variables of φ such that $\varphi \Leftrightarrow \varphi_{QF}$.
- $T(\mathbb{R})$ admits **quantifier elimination**.
- The ***decidability*** of $T(\mathbb{R})$

Example

$$\begin{aligned} \exists x. ax^2 + bx + c = 0 \quad \Leftrightarrow \quad & a = b = c = 0 \lor \\ & (a = 0 \land b \neq 0) \lor \\ & (a \neq 0 \land b^2 - 4ac \geq 0) \end{aligned}$$

Quantifier Elimination

Quantifier Elimination Property

- A theory $\mathcal{T}$ is said to have **quantifier elimination property**, if for any formula φ in $\mathcal{T}$, there exists a quantifier-free formula φ_{QF} which only contains *free* variables of φ such that $\varphi \Leftrightarrow \varphi_{QF}$.
- $T(\mathbb{R})$ admits **quantifier elimination**.
- The ***decidability*** of $T(\mathbb{R})$

Example

$$\begin{aligned}\exists x. ax^2 + bx + c = 0 \quad \Leftrightarrow \quad & a = b = c = 0 \vee \\ & (a = 0 \wedge b \neq 0) \vee \\ & (a \neq 0 \wedge b^2 - 4ac \geq 0)\end{aligned}$$

Quantifier Elimination

Quantifier Elimination Property

- A theory $\mathcal{T}$ is said to have **quantifier elimination property**, if for any formula φ in $\mathcal{T}$, there exists a quantifier-free formula φ_{QF} which only contains *free* variables of φ such that $\varphi \Leftrightarrow \varphi_{QF}$.
- $T(\mathbb{R})$ admits **quantifier elimination**.
- The ***decidability*** of $T(\mathbb{R})$

Example

$$\begin{aligned}\exists x. ax^2 + bx + c = 0 \quad \Leftrightarrow \quad & a = b = c = 0 \vee \\ & (a = 0 \wedge b \neq 0) \vee \\ & (a \neq 0 \wedge b^2 - 4ac \geq 0)\end{aligned}$$

Quantifier Elimination (Cont'd)

Semi-algrbraic Set

- A subset $A \subseteq \mathbb{R}^n$ is called a ***semi-algebraic set*** (SAS), if there exists a QFF $\phi \in T(\mathbb{R})$, such that $A = \{\mathbf{x} \in \mathbb{R}^n \mid \phi(\mathbf{x}) \text{ is true}\}$.
- SASs are closed under common set operations:
  - $\mathcal{A}(\phi_1) \cap \mathcal{A}(\phi_2) = \mathcal{A}(\varphi_1 \wedge \varphi_2)$;
  - $\mathcal{A}(\phi_1) \cup \mathcal{A}(\phi_2) = \mathcal{A}(\varphi_1 \vee \phi_2)$;
  - $\mathcal{A}(\phi_1)^c = \mathcal{A}(\neg\phi_1)$;
  - $\mathcal{A}(\phi_1) \setminus \mathcal{A}(\phi_2) = \mathcal{A}(\phi_1) \cap \mathcal{A}(\phi_2)^c = \mathcal{A}(\phi_1 \wedge \neg\phi_2)$.
- Any SAS can be represented by a QFF in the form of $\phi(\mathbf{x}) \mathrel{\widehat{=}} \bigvee_{k=1}^{K} \bigwedge_{j=1}^{J_k} p_{kj}(\mathbf{x}) \triangleright 0$, where $p_{kj}(\mathbf{x}) \in \mathbb{Q}[\mathbf{x}]$ and $\triangleright \in \{\geq, >\}$.

Semi-algebraic Template

A ***semi-algebraic template*** with degree d is of the form

$$\phi(\mathbf{u}, \mathbf{x}) \mathrel{\widehat{=}} \vee_{k=1}^{K} \wedge_{j=1}^{J_k} p_{kj}(\mathbf{u}_{kj}, \mathbf{x}) \triangleright 0.$$

Quantifier Elimination (Cont'd)

Semi-algrbraic Set

- A subset $A \subseteq \mathbb{R}^n$ is called a ***semi-algebraic set*** (SAS), if there exists a QFF $\phi \in T(\mathbb{R})$, such that $A = \{\mathbf{x} \in \mathbb{R}^n \mid \phi(\mathbf{x}) \text{ is true}\}$.
- SASs are closed under common set operations:
  - $\mathcal{A}(\phi_1) \cap \mathcal{A}(\phi_2) = \mathcal{A}(\varphi_1 \wedge \varphi_2)$;
  - $\mathcal{A}(\phi_1) \cup \mathcal{A}(\phi_2) = \mathcal{A}(\varphi_1 \vee \phi_2)$;
  - $\mathcal{A}(\phi_1)^c = \mathcal{A}(\neg\phi_1)$;
  - $\mathcal{A}(\phi_1) \setminus \mathcal{A}(\phi_2) = \mathcal{A}(\phi_1) \cap \mathcal{A}(\phi_2)^c = \mathcal{A}(\phi_1 \wedge \neg\phi_2)$.
- Any SAS can be represented by a QFF in the form of $\phi(\mathbf{x}) \widehat{=} \bigvee_{k=1}^{K} \bigwedge_{j=1}^{J_k} p_{kj}(\mathbf{x}) \triangleright 0$, where $p_{kj}(\mathbf{x}) \in \mathbb{Q}[\mathbf{x}]$ and $\triangleright \in \{\geq, >\}$.

Semi-algebraic Template

A ***semi-algebraic template*** with degree d is of the form

$$\phi(\mathbf{u}, \mathbf{x}) \widehat{=} \vee_{k=1}^{K} \wedge_{j=1}^{J_k} p_{kj}(\mathbf{u}_{kj}, \mathbf{x}) \triangleright 0.$$

Quantifier Elimination (Cont'd)

Semi-algrbraic Set

- A subset $A \subseteq \mathbb{R}^n$ is called a ***semi-algebraic set*** (SAS), if there exists a QFF $\phi \in T(\mathbb{R})$, such that $A = \{\mathbf{x} \in \mathbb{R}^n \mid \phi(\mathbf{x}) \text{ is true}\}$.
- SASs are closed under common set operations:
  - $\mathcal{A}(\phi_1) \cap \mathcal{A}(\phi_2) = \mathcal{A}(\varphi_1 \wedge \varphi_2)$;
  - $\mathcal{A}(\phi_1) \cup \mathcal{A}(\phi_2) = \mathcal{A}(\varphi_1 \vee \phi_2)$;
  - $\mathcal{A}(\phi_1)^c = \mathcal{A}(\neg\phi_1)$;
  - $\mathcal{A}(\phi_1) \setminus \mathcal{A}(\phi_2) = \mathcal{A}(\phi_1) \cap \mathcal{A}(\phi_2)^c = \mathcal{A}(\phi_1 \wedge \neg\phi_2)$.
- Any SAS can be represented by a QFF in the form of $\phi(\mathbf{x}) \widehat{=} \bigvee_{k=1}^{K} \bigwedge_{j=1}^{J_k} p_{kj}(\mathbf{x}) \triangleright 0$, where $p_{kj}(\mathbf{x}) \in \mathbb{Q}[\mathbf{x}]$ and $\triangleright \in \{\geq, >\}$.

Semi-algebraic Template

A ***semi-algebraic template*** with degree d is of the form

$$\phi(\mathbf{u}, \mathbf{x}) \widehat{=} \vee_{k=1}^{K} \wedge_{j=1}^{J_k} p_{kj}(\mathbf{u}_{kj}, \mathbf{x}) \triangleright 0.$$

Quantifier Elimination (Cont'd)

Semi-algrbraic Set

- A subset $A \subseteq \mathbb{R}^n$ is called a ***semi-algebraic set*** (SAS), if there exists a QFF $\phi \in T(\mathbb{R})$, such that $A = \{\mathbf{x} \in \mathbb{R}^n \mid \phi(\mathbf{x}) \text{ is true}\}$.
- SASs are closed under common set operations:
  - $\mathcal{A}(\phi_1) \cap \mathcal{A}(\phi_2) = \mathcal{A}(\varphi_1 \wedge \varphi_2)$;
  - $\mathcal{A}(\phi_1) \cup \mathcal{A}(\phi_2) = \mathcal{A}(\varphi_1 \vee \phi_2)$;
  - $\mathcal{A}(\phi_1)^c = \mathcal{A}(\neg\phi_1)$;
  - $\mathcal{A}(\phi_1) \setminus \mathcal{A}(\phi_2) = \mathcal{A}(\phi_1) \cap \mathcal{A}(\phi_2)^c = \mathcal{A}(\phi_1 \wedge \neg\phi_2)$.
- Any SAS can be represented by a QFF in the form of $\phi(\mathbf{x}) \widehat{=} \bigvee_{k=1}^{K} \bigwedge_{j=1}^{J_k} p_{kj}(\mathbf{x}) \triangleright 0$, where $p_{kj}(\mathbf{x}) \in \mathbb{Q}[\mathbf{x}]$ and $\triangleright \in \{\geq, >\}$.

Semi-algebraic Template

A ***semi-algebraic template*** with degree d is of the form

$$\phi(\mathbf{u}, \mathbf{x}) \widehat{=} \vee_{k=1}^{K} \wedge_{j=1}^{J_k} p_{kj}(\mathbf{u}_{kj}, \mathbf{x}) \triangleright 0.$$

Quantifier Elimination (Cont'd)

Survey of QE Algorithms

- **Tarski's algorithm** [Tarski 51]: the first one, but its complexity is nonelementary, impratical, simplified by Seidenberg [Seidenberg 54].
- **Collins' algorithm** [Collins 76]: based on **cylindrical algebraic decomposition (CAD)**, **double exponential** in the number of variables, improved by Hoon Hong [Hoon Hong 92] by combining with SAT engine **partial cylindrical algebraic decomposition (PCAD)**, implemented in many computer algebra tools, e.g., **QEBCAD, REDLOG, ...**
- **Ben-Or, Kozen and Reif's algorithm** [Ben-Or, Kozen&Reif 1986]: double exponential in the number of variables using sequential computation, single exponential using parallel computation, based on **Sturm sequence** and **Sturm Theorem**, some mistake.
- More efficient algorithms [Grigor'ev & Vorobjov 1988, Grigor'ev 1988], [Renegar 1989], [Heintz, Roy&Solerno, 1989], [Basu,Pollack&Roy, 1996]: mainly based on **Ben-Or, Kozen and Reif's work**, double exponential in the number of quantifier alternation, no implementation yet.

Quantifier Elimination (Cont'd)

Survey of QE Algorithms

- **Tarski's algorithm** [Tarski 51]: the first one, but its complexity is nonelementary, imprатical, simplified by Seidenberg [Seidenberg 54].
- **Collins' algorithm** [Collins 76]: based on **cylindrical algebraic decomposition (CAD)**, **double exponential** in the number of variables, improved by Hoon Hong [Hoon Hong 92] by combining with SAT engine **partial cylindrical algebraic decomposition (PCAD)**, implemented in many computer algebra tools, e.g., **QEBCAD, REDLOG, ...**
- **Ben-Or, Kozen and Reif's algorithm** [Ben-Or, Kozen&Reif 1986]: double exponential in the number of variables using sequential computation, single exponential using parallel computation, based on **Sturm sequence** and **Sturm Theorem**, some mistake.
- More efficient algorithms [Grigor'ev & Vorobjov 1988, Grigor'ev 1988], [Renegar 1989], [Heintz, Roy&Solerno, 1989], [Basu,Pollack&Roy, 1996]: mainly based on **Ben-Or, Kozen and Reif's work**, double exponential in the number of quantifier alternation, no implementation yet.

Quantifier Elimination (Cont'd)

Survey of QE Algorithms

- **Tarski's algorithm** [Tarski 51]: the first one, but its complexity is nonelementary, impratical, simplified by Seidenberg [Seidenberg 54].
- **Collins' algorithm** [Collins 76]: based on **cylindrical algebraic decomposition (CAD)**, **double exponential** in the number of variables, improved by Hoon Hong [Hoon Hong 92] by combining with SAT engine **partial cylindrical algebraic decomposition (PCAD)**, implemented in many computer algebra tools, e.g., **QEBCAD, REDLOG, ...**
- **Ben-Or, Kozen and Reif's algorithm** [Ben-Or, Kozen&Reif 1986]: double exponential in the number of variables using sequential computation, single exponential using parallel computation, based on **Sturm sequence** and **Sturm Theorem**, some mistake.
- More efficient algorithms [Grigor'ev & Vorobjov 1988, Grigor'ev 1988], [Renegar 1989], [Heintz, Roy&Solerno, 1989], [Basu,Pollack&Roy, 1996]: mainly based on **Ben-Or, Kozen and Reif's work**, double exponential in the number of quantifier alternation, no implementation yet.

Quantifier Elimination (Cont'd)

Survey of QE Algorithms

- **Tarski's algorithm** [Tarski 51]: the first one, but its complexity is nonelementary, imprattical, simplified by Seidenberg [Seidenberg 54].
- **Collins' algorithm** [Collins 76]: based on **cylindrical algebraic decomposition (CAD)**, **double exponential** in the number of variables, improved by Hoon Hong [Hoon Hong 92] by combining with SAT engine **partial cylindrical algebraic decomposition (PCAD)**, implemented in many computer algebra tools, e.g., **QEBCAD, REDLOG, ...**
- **Ben-Or, Kozen and Reif's algorithm** [Ben-Or, Kozen&Reif 1986]: double exponential in the number of variables using sequential computation, single exponential using parallel computation, based on **Sturm sequence** and **Sturm Theorem**, some mistake.
- More efficient algorithms [Grigor'ev & Vorobjov 1988, Grigor'ev 1988], [Renegar 1989], [Heintz, Roy&Solerno, 1989], [Basu,Pollack&Roy, 1996]: mainly based on **Ben-Or, Kozen and Reif's work**, double exponential in the number of quantifier alternation, no implementation yet.

Outline

Continuous Dynamical Systems

- A **continuous dynamical systems (CDS)** is of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a ***vector field***.

- If $\mathbf{f}$ in (1) satisfies **local Lipschitz condition**, then given $\mathbf{x}_0 \in \mathbb{R}^n$, there exists a unique solution $\mathbf{x}(\mathbf{x}_0; t) : (a, b) \to \mathbb{R}^n$ such that $\mathbf{x}(\mathbf{x}_0; 0) = \mathbf{x}_0$ and $\forall t \in (a, b). \frac{\mathrm{d}\mathbf{x}(\mathbf{x}_0;t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(\mathbf{x}_0; t))$.
- If $\mathbf{f}$ in (1) satisfies **global Lipschitz condition**, then the **existence, uniqueness** and **completeness** of solutions to (1) can be guaranteed.
- The k-th **Lie derivatives** $L_{\mathbf{f}}^k \sigma : \mathbb{R}^n \to \mathbb{R}$ of σ along $\mathbf{f}$ is defined by:
  - $L_{\mathbf{f}}^0 \sigma(\mathbf{x}) = \sigma(\mathbf{x})$,
  - $L_{\mathbf{f}}^k \sigma(\mathbf{x}) = \left(\nabla L_{\mathbf{f}}^{k-1} \sigma(\mathbf{x}), \mathbf{f}(\mathbf{x})\right)$, for $k > 0$,

  where $\nabla \varrho(\mathbf{x}) \widehat{=} \left(\frac{\partial \varrho(\mathbf{x})}{\partial x_1}, \frac{\partial \varrho(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial \varrho(\mathbf{x})}{\partial x_n}\right)$ and $(\cdot, \cdot)$ is the *inner product* of two vectors.

Continuous Dynamical Systems

- A **continuous dynamical systems (CDS)** is of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a ***vector field***.

- If $\mathbf{f}$ in (1) satisfies **local Lipschitz condition**, then given $\mathbf{x}_0 \in \mathbb{R}^n$, there exists a unique solution $\mathbf{x}(\mathbf{x}_0; t) : (a, b) \to \mathbb{R}^n$ such that $\mathbf{x}(\mathbf{x}_0; 0) = \mathbf{x}_0$ and $\forall t \in (a, b). \frac{\mathrm{d}\mathbf{x}(\mathbf{x}_0;t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(\mathbf{x}_0; t))$.
- If $\mathbf{f}$ in (1) satisfies **global Lipschitz condition**, then the **existence, uniqueness** and **completeness** of solutions to (1) can be guaranteed.
- The k-th **Lie derivatives** $L_{\mathbf{f}}^k \sigma : \mathbb{R}^n \to \mathbb{R}$ of σ along $\mathbf{f}$ is defined by:
  - $L_{\mathbf{f}}^0 \sigma(\mathbf{x}) = \sigma(\mathbf{x})$,
  - $L_{\mathbf{f}}^k \sigma(\mathbf{x}) = \left(\nabla L_{\mathbf{f}}^{k-1} \sigma(\mathbf{x}), \mathbf{f}(\mathbf{x})\right)$, for $k > 0$,

  where $\nabla \varrho(\mathbf{x}) \widehat{=} \left(\frac{\partial \varrho(\mathbf{x})}{\partial x_1}, \frac{\partial \varrho(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial \varrho(\mathbf{x})}{\partial x_n}\right)$ and $(\cdot, \cdot)$ is the *inner product* of two vectors.

Continuous Dynamical Systems

- A **continuous dynamical systems (CDS)** is of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a ***vector field***.

- If $\mathbf{f}$ in (1) satisfies **local Lipschitz condition**, then given $\mathbf{x}_0 \in \mathbb{R}^n$, there exists a unique solution $\mathbf{x}(\mathbf{x}_0; t) : (a, b) \to \mathbb{R}^n$ such that $\mathbf{x}(\mathbf{x}_0; 0) = \mathbf{x}_0$ and $\forall t \in (a, b). \frac{\mathrm{d}\mathbf{x}(\mathbf{x}_0; t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(\mathbf{x}_0; t))$.
- If $\mathbf{f}$ in (1) satisfies **global Lipschitz condition**, then the **existence, uniqueness** and **completeness** of solutions to (1) can be guaranteed.
- The k-th **Lie derivatives** $L_{\mathbf{f}}^k \sigma : \mathbb{R}^n \to \mathbb{R}$ of σ along $\mathbf{f}$ is defined by:
  - $L_{\mathbf{f}}^0 \sigma(\mathbf{x}) = \sigma(\mathbf{x})$,
  - $L_{\mathbf{f}}^k \sigma(\mathbf{x}) = \left(\nabla L_{\mathbf{f}}^{k-1} \sigma(\mathbf{x}), \mathbf{f}(\mathbf{x}) \right)$, for $k > 0$,

  where $\nabla \varrho(\mathbf{x}) \widehat{=} \left(\frac{\partial \varrho(\mathbf{x})}{\partial x_1}, \frac{\partial \varrho(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial \varrho(\mathbf{x})}{\partial x_n} \right)$ and $(\cdot, \cdot)$ is the *inner product* of two vectors.

Continuous Dynamical Systems

- A **continuous dynamical systems (CDS)** is of the form
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{1}$$
where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a ***vector field***.
- If $\mathbf{f}$ in (1) satisfies **local Lipschitz condition**, then given $\mathbf{x}_0 \in \mathbb{R}^n$, there exists a unique solution $\mathbf{x}(\mathbf{x}_0; t) : (a, b) \to \mathbb{R}^n$ such that $\mathbf{x}(\mathbf{x}_0; 0) = \mathbf{x}_0$ and $\forall t \in (a, b). \frac{\mathrm{d}\mathbf{x}(\mathbf{x}_0;t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(\mathbf{x}_0; t))$.
- If $\mathbf{f}$ in (1) satisfies **global Lipschitz condition**, then the **existence, uniqueness** and **completeness** of solutions to (1) can be guaranteed.
- The k-th **Lie derivatives** $L_{\mathbf{f}}^k\sigma : \mathbb{R}^n \to \mathbb{R}$ of σ along $\mathbf{f}$ is defined by:
  - $L_{\mathbf{f}}^0\sigma(\mathbf{x}) = \sigma(\mathbf{x})$,
  - $L_{\mathbf{f}}^k\sigma(\mathbf{x}) = \left(\nabla L_{\mathbf{f}}^{k-1}\sigma(\mathbf{x}), \mathbf{f}(\mathbf{x})\right)$, for $k > 0$,

  where $\nabla\varrho(\mathbf{x}) \widehat{=} \left(\frac{\partial\varrho(\mathbf{x})}{\partial x_1}, \frac{\partial\varrho(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial\varrho(\mathbf{x})}{\partial x_n}\right)$ and $(\cdot, \cdot)$ is the *inner product* of two vectors.

Outline

Hybrid Automaton

A **hybrid automaton** (HA) is a system $\mathcal{H} \mathrel{\widehat{=}} (Q, X, f, D, E, G, R, \Xi)$, where

- $Q = \{q_1, \ldots, q_m\}$ is a finite set of modes;
- $X = \{x_1, \ldots, x_n\}$ is a finite set of continuous state variables, with $\mathbf{x} = (x_1, \ldots, x_n)$ ranging over $\mathbb{R}^n$; $Q \times \mathbb{R}^n$ is the state space of $\mathcal{H}$;
- $f : Q \to (\mathbb{R}^n \to \mathbb{R}^n)$ assigns to each mode $q \in Q$ a vector field $\mathbf{f}_q$;
- $D : Q \to 2^{\mathbb{R}^n}$ assigns to each mode $q \in Q$ a domain $D_q \subseteq \mathbb{R}^n$;
- $E \subseteq Q \times Q$ is a set of discrete transitions;
- $G : E \to 2^{\mathbb{R}^n}$ assigns to each transition $e \in E$ a switching guard $G_e \subseteq \mathbb{R}^n$.
- R assigns to each transition $e \in E$ a reset function $R_e \colon \mathbb{R}^n \to \mathbb{R}^n$;
- Ξ assigns to each $q \in Q$ a set of initial states $\Xi_q \subseteq \mathbb{R}^n$.

Hybrid Trajectories Accepted by HA [Tomlin et al. 00]

Definition (Hybrid Time Set)

A hybrid time set is a sequence of time intervals $\tau = \{I_i\}_{i=0}^{N}$ (N can be ∞) s.t.

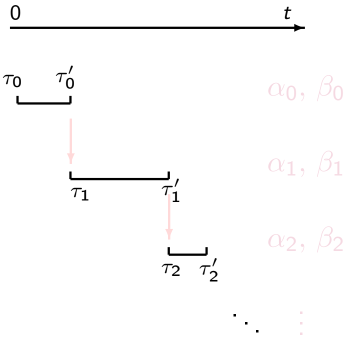
- $I_i = [\tau_i, \tau_i']$ with $\tau_i \leq \tau_i' = \tau_{i+1}$ for all $i < N$;
- if $N < \infty$, then $I_N = [\tau_N, \tau_N'\rangle$ is a right-closed or right-open nonempty interval (τ_N' may be ∞);
- $\tau_0 = 0$.

Hybrid Trajectories Accepted by HA [Tomlin et al. 00]

Definition (Hybrid Time Set)

A hybrid time set is a sequence of time intervals $\tau = \{I_i\}_{i=0}^{N}$ (N can be ∞) s.t.

- $I_i = [\tau_i, \tau_i']$ with $\tau_i \leq \tau_i' = \tau_{i+1}$ for all $i < N$;
- if $N < \infty$, then $I_N = [\tau_N, \tau_N'\rangle$ is a right-closed or right-open nonempty interval (τ_N' may be ∞);
- $\tau_0 = 0$.

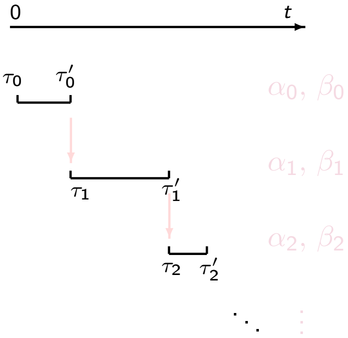


Hybrid Trajectories Accepted by HA [Tomlin et al. 00]

Definition (Hybrid Trajectory)

A hybrid trajectory is a triple $\omega = (\tau, \alpha, \beta)$, where $\tau = \{I_i\}_{i=0}^{N}$ is a hybrid time set and $\alpha = \{\alpha_i : I_i \to Q\}$ and $\beta = \{\beta_i : I_i \to \mathbb{R}^n\}$ are two sequences of functions satisfying

1. Initial condition: $\alpha_0[0] = q_0$ and $\beta_0[0] = \mathbf{x}_0$;
2. Discrete transition: for all $i < \langle \tau \rangle$, $e = (\alpha_i(\tau_i'), \alpha_{i+1}(\tau_{i+1})) \in E$, $\beta_i(\tau_i') \in G_e$ and $\beta_{i+1}(\tau_{i+1}) = R_e(\beta_i(\tau_i'))$;
3. Continuous evolution: for all $i \leq \langle \tau \rangle$ with $\tau_i < \tau_i'$, if $q = \alpha_i(\tau_i)$, then
   (1) for all $t \in I_i$, $\alpha_i(t) = q$,
   (2) $\beta_i(t)$ is the solution to the differential equation $\dot{\mathbf{x}} = \mathbf{f}_q(\mathbf{x})$ over I_i with initial value $\beta_i(\tau_i)$, and
   (3) for all $t \in [\tau_i, \tau_i')$, $\beta_i(t) \in D_q$.

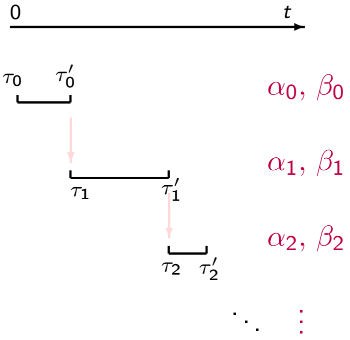


Hybrid Trajectories Accepted by HA [Tomlin et al. 00]

Definition (Hybrid Trajectory)

A hybrid trajectory is a triple $\omega = (\tau, \alpha, \beta)$, where $\tau = \{I_i\}_{i=0}^{N}$ is a hybrid time set and $\alpha = \{\alpha_i : I_i \to Q\}$ and $\beta = \{\beta_i : I_i \to \mathbb{R}^n\}$ are two sequences of functions satisfying

1. Initial condition: $\alpha_0[0] = q_0$ and $\beta_0[0] = \mathbf{x}_0$;
2. Discrete transition: for all $i < \langle\tau\rangle$, $e = (\alpha_i(\tau_i'), \alpha_{i+1}(\tau_{i+1})) \in E$, $\beta_i(\tau_i') \in G_e$ and $\beta_{i+1}(\tau_{i+1}) = R_e(\beta_i(\tau_i'))$;
3. Continuous evolution: for all $i \leq \langle\tau\rangle$ with $\tau_i < \tau_i'$, if $q = \alpha_i(\tau_i)$, then
   (1) for all $t \in I_i$, $\alpha_i(t) = q$,
   (2) $\beta_i(t)$ is the solution to the differential equation $\dot{\mathbf{x}} = \mathbf{f}_q(\mathbf{x})$ over I_i with initial value $\beta_i(\tau_i)$, and
   (3) for all $t \in [\tau_i, \tau_i')$, $\beta_i(t) \in D_q$.

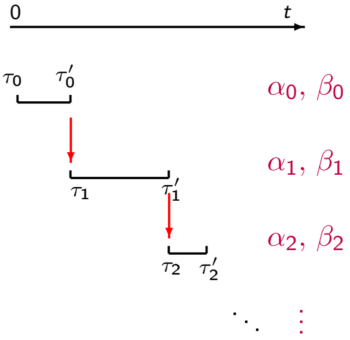


Hybrid Trajectories Accepted by HA [Tomlin et al. 00]

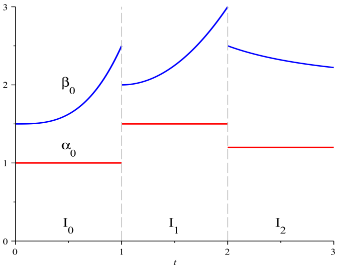
Definition (Hybrid Trajectory)

A hybrid trajectory is a triple $\omega = (\tau, \alpha, \beta)$, where $\tau = \{I_i\}_{i=0}^{N}$ is a hybrid time set and $\alpha = \{\alpha_i : I_i \to Q\}$ and $\beta = \{\beta_i : I_i \to \mathbb{R}^n\}$ are two sequences of functions satisfying

1. Initial condition: $\alpha_0[0] = q_0$ and $\beta_0[0] = \mathbf{x}_0$;
2. Discrete transition: for all $i < \langle\tau\rangle$, $e = (\alpha_i(\tau_i'), \alpha_{i+1}(\tau_{i+1})) \in E$, $\beta_i(\tau_i') \in G_e$ and $\beta_{i+1}(\tau_{i+1}) = R_e(\beta_i(\tau_i'))$;
3. Continuous evolution: for all $i \leq \langle\tau\rangle$ with $\tau_i < \tau_i'$, if $q = \alpha_i(\tau_i)$, then
   (1) for all $t \in I_i$, $\alpha_i(t) = q$,
   (2) $\beta_i(t)$ is the solution to the differential equation $\dot{\mathbf{x}} = \mathbf{f}_q(\mathbf{x})$ over I_i with initial value $\beta_i(\tau_i)$, and
   (3) for all $t \in [\tau_i, \tau_i')$, $\beta_i(t) \in D_q$.



Hybrid Trajectories Accepted by HA (Cont'd) [Tomlin et al. 00]

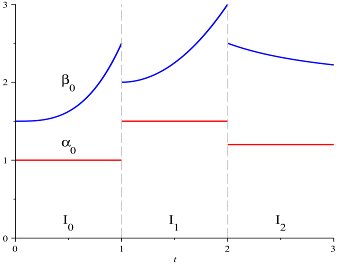


A hybrid trajectory (τ, α, β) is called *infinite* if

- $\langle \tau \rangle = N$ is ∞, or
- $\|\tau\| = \sum_{i=0}^{N} (\tau_i' - \tau_i)$ is ∞.

A hybrid automaton is called *non-blocking* if there is an infinite trajectory starting from any initial state (q_0, x_0), and *blocking* otherwise.

Hybrid Trajectories Accepted by HA (Cont'd) [Tomlin et al. 00]

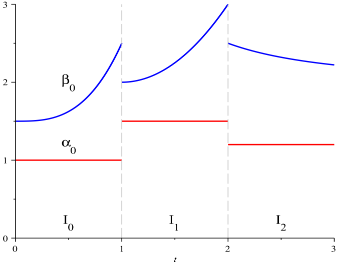


A hybrid trajectory (τ, α, β) is called *infinite* if

- $\langle \tau \rangle = N$ is ∞, or
- $\|\tau\| = \sum_{i=0}^{N}(\tau_i' - \tau_i)$ is ∞.

A hybrid automaton is called *non-blocking* if there is an infinite trajectory starting from any initial state (q_0, x_0), and *blocking* otherwise.

Hybrid Trajectories Accepted by HA (Cont'd) [Tomlin et al. 00]



A hybrid trajectory (τ, α, β) is called *infinite* if

- $\langle\tau\rangle = N$ is ∞, or
- $\|\tau\| = \sum_{i=0}^{N}(\tau_i' - \tau_i)$ is ∞.

A hybrid automaton is called *non-blocking* if there is an infinite trajectory starting from any initial state $(q_0, \mathbf{x}_0)$, and *blocking* otherwise.

Reachable Set of HA

Definition (Reachable Set)

Given an HA $\mathcal{H}$, the **reachable set** $\mathcal{R}_{\mathcal{H}}$ of $\mathcal{H}$ consists of those $(q, \mathbf{x})$ for which there exists a finite sequence

$$(q_0, \mathbf{x}_0), (q_1, \mathbf{x}_1), \ldots, (q_l, \mathbf{x}_l)$$

such that $(q_0, \mathbf{x}_0) \in \Xi_{\mathcal{H}}$, $(q_l, \mathbf{x}_l) = (q, \mathbf{x})$, and for any $0 \leq i \leq l-1$, one of the following two conditions holds:

- **(Discrete Jump):** $e = (q_i, q_{i+1}) \in E$, $\mathbf{x}_i \in G_e$ and $\mathbf{x}_{i+1} = R_e(\mathbf{x}_i)$; or
- **(Continuous Evolution):** $q_i = q_{i+1}$, and there exists a $\delta \geq 0$ s.t. the solution $\mathbf{x}(\mathbf{x}_i; t)$ to $\dot{\mathbf{x}} = \mathbf{f}_{q_i}$ satisfies
  - $\mathbf{x}(\mathbf{x}_i; t) \in D_{q_i}$ for all $t \in [0, \delta]$; and
  - $\mathbf{x}(\mathbf{x}_i; \delta) = \mathbf{x}_{i+1}$.

Outline

Continuous vs Global Invariants

Note that

- **Hybrid systems** consists of a set of CDSs, a set of transitions between these CDSs, and a transition may be equipped with a guard and reset
- **Invariant** plays a key role in analysis, verification, synthesis of hybrid systems
- **Global invariant** keeps invariant during continuous and discrete evolutions
- **Continuous invariant** keeps invariant in a mode
- Interplay between global and continuous invariant
- Both can be reduced to constraint solving
- Continuous invariant (differential invariant) generation is more complicated

Global Invariant

Definition (Global Invariant)

An invariant of an HA $\mathcal{H}$ maps to each $q \in Q$ a subset $I_q \subseteq \mathbb{R}^n$, such that for all $(q, \mathbf{x}) \in \mathcal{R}_{\mathcal{H}}$ (the reachable set), we have $\mathbf{x} \in I_q$.

Definition (Inductive Invariant)

Given an HA $\mathcal{H}$, an **inductive invariant** maps to each $q \in Q$ a subset $I_q \subseteq \mathbb{R}^n$, such that the following conditions are satisfied:

1. $\Xi_q \subseteq I_q$ for all $q \in Q$;
2. for any $e = (q, q') \in E$, if $\mathbf{x} \in I_q \cap G_e$, then $\mathbf{x}' = R_e(\mathbf{x}) \in I_{q'}$;
3. for any $q \in Q$ and any $\mathbf{x}_0 \in I_q$, if there exists a $\delta \geq 0$ s.t. the solution $\mathbf{x}(\mathbf{x}_0; t)$ to $\dot{\mathbf{x}} = \mathbf{f}_q$ satisfies: (i) $\mathbf{x}(\mathbf{x}_0; \delta) = \mathbf{x}'$; and (ii) $\mathbf{x}(\mathbf{x}_0; t) \in D_q$ for all $t \in [0, \delta]$, then $\mathbf{x}' \in I_q$.

Global Invariant

Definition (Global Invariant)

An invariant of an HA $\mathcal{H}$ maps to each $q \in Q$ a subset $I_q \subseteq \mathbb{R}^n$, such that for all $(q, \mathbf{x}) \in \mathcal{R}_{\mathcal{H}}$ (the reachable set), we have $\mathbf{x} \in I_q$.

Definition (Inductive Invariant)

Given an HA $\mathcal{H}$, an **inductive invariant** maps to each $q \in Q$ a subset $I_q \subseteq \mathbb{R}^n$, such that the following conditions are satisfied:

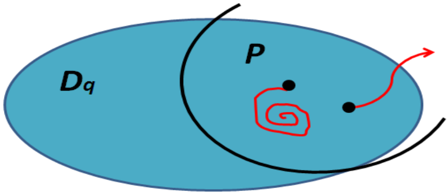
1. $\Xi_q \subseteq I_q$ for all $q \in Q$;
2. for any $e = (q, q') \in E$, if $\mathbf{x} \in I_q \cap G_e$, then $\mathbf{x}' = R_e(\mathbf{x}) \in I_{q'}$;
3. for any $q \in Q$ and any $\mathbf{x}_0 \in I_q$, if there exists a $\delta \geq 0$ s.t. the solution $\mathbf{x}(\mathbf{x}_0; t)$ to $\dot{\mathbf{x}} = \mathbf{f}_q$ satisfies: (i) $\mathbf{x}(\mathbf{x}_0; \delta) = \mathbf{x}'$; and (ii) $\mathbf{x}(\mathbf{x}_0; t) \in D_q$ for all $t \in [0, \delta]$, then $\mathbf{x}' \in I_q$.

Continuous Invariant

Definition (Continuous Invariant see also [Platzer & Clarke 08] **)**

Given $(D_q, \mathbf{f}_q)$, we call $P \subseteq \mathbb{R}^n$ a **continuous invariant** of $(D_q, \mathbf{f}_q)$ if for all $\mathbf{x}_0 \in P$ and all $T \geq 0$,

$$(\forall t \in [0, T].\, \mathbf{x}(t) \in D_q) \implies (\forall t \in [0, T].\, \mathbf{x}(t) \in P) \ .$$



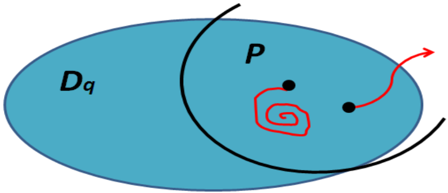
- A continuous invariant of a PDS is called a *semi-algebraic invariant* (SAI) if it is a semi-algebraic set.

Continuous Invariant

Definition (Continuous Invariant see also [Platzer & Clarke 08] **)**

Given $(D_q, \mathbf{f}_q)$, we call $P \subseteq \mathbb{R}^n$ a **continuous invariant** of $(D_q, \mathbf{f}_q)$ if for all $\mathbf{x}_0 \in P$ and all $T \geq 0$,

$$(\forall t \in [0, T].\, \mathbf{x}(t) \in D_q) \implies (\forall t \in [0, T].\, \mathbf{x}(t) \in P) \ .$$



- A continuous invariant of a PDS is called a ***semi-algebraic invariant*** (SAI) if it is a semi-algebraic set.

Related Work

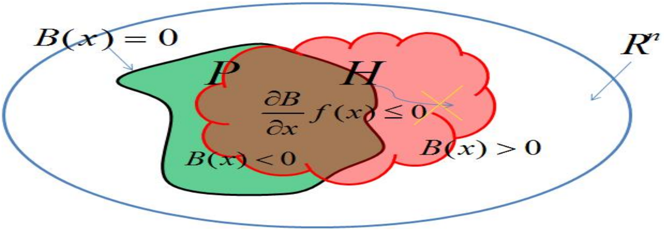
- **Barrier-certificate** [Prajna&Jadbadbaie 2004, Plazer&Clarke 2008]
  - Basic idea: Let $\mathcal{D} = \{\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})\}$ and $H = \{h(\mathbf{x}) \geq 0\}$. A function $B : \mathbb{R}^n \to \mathbb{R}$ is a barrier certificate if it is differentiable and satisfying

$$\forall \mathbf{x} \in H.\ \frac{\partial B}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}) \leq 0.$$

or

$$\forall \mathbf{x} \in H(B(x) = 0 \Rightarrow \frac{\partial B}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}) < 0).$$

Let $P := \{\mathbf{x} \mid B(\mathbf{x}) \leq 0\}$. Then P is an invariant of $(\mathcal{D}, H)$.



Related Work (Cont'd)

- **Boundary method** [Taly, Gulwani&Tiwari, VMCAI 2009]

  Let $\mathcal{D} = \{\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})\}$ and $H = \{h(\mathbf{x}) \geq 0\}$. If $P := \{\mathbf{x} \mid p(\mathbf{x}) \geq 0\}$ has the following property: For each $\mathbf{x}$ s.t. $p(\mathbf{x}) = 0$, there is a $\delta > 0$ s.t.

  $$\forall \mathbf{y} : (p(\mathbf{y}) = 0 \wedge \|\mathbf{y} - \mathbf{x}\| < \delta \Rightarrow L_{\mathbf{f}} p(\mathbf{y}) \geq 0 \wedge \frac{\partial p}{\partial \mathbf{y}} \neq \mathbf{0}),$$

  then P is an invariant of $(\mathcal{D}, H)$.

  - It imposes a strong assumption on the boundary.

- **Ideal fixed point method** [Sankaranarayanan, HSCC 2010]
  - Basic idea: If an ideal $\mathcal{I} \subseteq \mathcal{R}[\mathbf{x}]$ has the property:
    1. $(\forall p \in \mathcal{I}, \mathbf{x} \in H) p(\mathbf{x}) = 0$,
    2. $(\forall p \in \mathcal{I}), L_{\mathbf{f}} p \in \mathcal{I}$;

    then $P := \{\mathbf{x} \mid p(\mathbf{x}) = 0, \forall p \in \mathcal{I}\}$ is an invariant of $(\mathcal{D}, H)$.
  - It cannot cope with invariants as general semi-algebraic sets.

Related Work (Cont'd)

- **Boundary method** [Taly, Gulwani&Tiwari, VMCAI 2009]

  Let $\mathcal{D} = \{\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})\}$ and $H = \{h(\mathbf{x}) \geq 0\}$. If $P := \{\mathbf{x} \mid p(\mathbf{x}) \geq 0\}$ has the following property: For each $\mathbf{x}$ s.t. $p(\mathbf{x}) = 0$, there is a $\delta > 0$ s.t.

  $$\forall \mathbf{y} : (p(\mathbf{y}) = 0 \wedge \|\mathbf{y} - \mathbf{x}\| < \delta \Rightarrow L_{\mathbf{f}}\, p(\mathbf{y}) \geq 0 \wedge \frac{\partial p}{\partial \mathbf{y}} \neq \mathbf{0}),$$

  then P is an invariant of $(\mathcal{D}, H)$.

  - It imposes a strong assumption on the boundary.

- **Ideal fixed point method** [Sankaranarayanan, HSCC 2010]
  - Basic idea: If an ideal $\mathcal{I} \subseteq \mathcal{R}[\mathbf{x}]$ has the property:
    1. $(\forall p \in \mathcal{I}, \mathbf{x} \in H) p(\mathbf{x}) = 0$,
    2. $(\forall p \in \mathcal{I}), L_{\mathbf{f}}\, p \in \mathcal{I}$;

    then $P := \{\mathbf{x} \mid p(\mathbf{x}) = 0, \forall p \in \mathcal{I}\}$ is an invariant of $(\mathcal{D}, H)$.
  - It cannot cope with invariants as general semi-algebraic sets.

Related Work (Cont'd)

Open Problem

- **Open problem** [Sankaranarayanan, HSCC 2010, Taly&Tiwari, FSTTCS 2009]: **Can we find a complete method to generate all semi-algebraic invariants of a polynomial dynamical system?**
- We addressed this problem and gave an affirmative answer in [Liu, Zhan&Zhao 2011].

Outline

Basic Idea

- Let $(D, \mathbf{f})$ be a PDS, $\mathbf{x}(t)$ is a trajectory of $(D, \mathbf{f})$ from $\mathbf{x}_0$, and $P \widehat{=} p(\mathbf{x}) \geq 0$. Then P be a differential invariant of $(D, \mathbf{f})$ iff

$$\forall \mathbf{x}_0 \in \partial P \cap D, \exists \epsilon > 0, \forall t \in [0, \epsilon]. p(\mathbf{x}(t)) \geq 0 \tag{2}$$

- $p(\mathbf{x}(t))$'s Taylor's expansion at $t = 0$

$$p(\mathbf{x}(t)) = L_{\mathbf{f}}^1 p(\mathbf{x}_0).t + L_{\mathbf{f}}^2 p(\mathbf{x}_0).\frac{t^2}{2!} + \cdots L_{\mathbf{f}}^i p(\mathbf{x}_0).\frac{t^i}{i!} + \cdots$$

- (2) holds iff
  1. either for all $i \geq 0, L_{\mathbf{f}}^i p(\mathbf{x}_0) = 0$
  2. or there is some $k > i \geq 0$, such that $L_{\mathbf{f}}^i p(\mathbf{x}_0) = 0$ and $L_{\mathbf{f}}^k p(\mathbf{x}_0) > 0$.
- The *pointwise rank* of p with respect to $\mathbf{f}$ as the function $\gamma_{p,\mathbf{f}} : \mathbb{R}^n \to \mathbb{N} \cup \{\infty\}$ defined by

$$\gamma_{p,\mathbf{f}}(\mathbf{x}) = \min\{k \in \mathbb{N} \mid L_{\mathbf{f}}^k p(\mathbf{x}) \neq 0\}$$

  if such k exists, and $\gamma_{p,\mathbf{f}}(\mathbf{x}) = \infty$ otherwise.

Basic Idea

- Let $(D, \mathbf{f})$ be a PDS, $\mathbf{x}(t)$ is a trajectory of $(D, \mathbf{f})$ from $\mathbf{x}_0$, and $P \widehat{=} p(\mathbf{x}) \geq 0$. Then P be a differential invariant of $(D, \mathbf{f})$ iff

$$\forall \mathbf{x}_0 \in \partial P \cap D, \exists \epsilon > 0, \forall t \in [0, \epsilon]. p(\mathbf{x}(t)) \geq 0 \tag{2}$$

- $p(\mathbf{x}(t))$'s **Taylor's expansion** at $t = 0$

$$p(\mathbf{x}(t)) = L^1_{\mathbf{f}} p(\mathbf{x}_0).t + L^2_{\mathbf{f}} p(\mathbf{x}_0).\frac{t^2}{2!} + \cdots L^i_{\mathbf{f}} p(\mathbf{x}_0).\frac{t^i}{i!} + \cdots$$

- (2) holds iff
  1. either for all $i \geq 0, L^i_{\mathbf{f}} p(\mathbf{x}_0) = 0$
  2. or there is some $k > i \geq 0$, such that $L^i_{\mathbf{f}} p(\mathbf{x}_0) = 0$ and $L^k_{\mathbf{f}} p(\mathbf{x}_0) > 0$.
- The *pointwise rank* of p with respect to $\mathbf{f}$ as the function $\gamma_{p,\mathbf{f}} : \mathbb{R}^n \to \mathbb{N} \cup \{\infty\}$ defined by

$$\gamma_{p,\mathbf{f}}(\mathbf{x}) = \min\{k \in \mathbb{N} \mid L^k_{\mathbf{f}} p(\mathbf{x}) \neq 0\}$$

if such k exists, and $\gamma_{p,\mathbf{f}}(\mathbf{x}) = \infty$ otherwise.

Basic Idea

- Let $(D, \mathbf{f})$ be a PDS, $\mathbf{x}(t)$ is a trajectory of $(D, \mathbf{f})$ from $\mathbf{x}_0$, and $P \widehat{=} p(\mathbf{x}) \geq 0$. Then P be a differential invariant of $(D, \mathbf{f})$ iff

$$\forall \mathbf{x}_0 \in \partial P \cap D, \exists \epsilon > 0, \forall t \in [0, \epsilon]. p(\mathbf{x}(t)) \geq 0 \tag{2}$$

- $p(\mathbf{x}(t))$'s **Taylor's expansion** at $t = 0$

$$p(\mathbf{x}(t)) = L_{\mathbf{f}}^1 p(\mathbf{x}_0).t + L_{\mathbf{f}}^2 p(\mathbf{x}_0).\frac{t^2}{2!} + \cdots L_{\mathbf{f}}^i p(\mathbf{x}_0).\frac{t^i}{i!} + \cdots$$

- (2) holds iff
  1. either for all $i \geq 0, L_{\mathbf{f}}^i p(\mathbf{x}_0) = 0$
  2. or there is some $k > i \geq 0$, such that $L_{\mathbf{f}}^i p(\mathbf{x}_0) = 0$ and $L_{\mathbf{f}}^k p(\mathbf{x}_0) > 0$.
- The *pointwise rank* of p with respect to $\mathbf{f}$ as the function $\gamma_{p,\mathbf{f}} : \mathbb{R}^n \to \mathbb{N} \cup \{\infty\}$ defined by

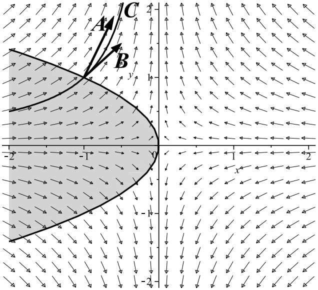
$$\gamma_{p,\mathbf{f}}(\mathbf{x}) = \min\{k \in \mathbb{N} \mid L_{\mathbf{f}}^k p(\mathbf{x}) \neq 0\}$$

if such k exists, and $\gamma_{p,\mathbf{f}}(\mathbf{x}) = \infty$ otherwise.

Example

Let $\mathbf{f} = (-x, y)$ and $p(x,y) = x + y^2$. Then

$$
\begin{aligned}
L_{\mathbf{f}}^{0} p(x,y) &= x + y^2 \\
L_{\mathbf{f}}^{1} p(x,y) &= -x + 2y^2 \\
L_{\mathbf{f}}^{2} p(x,y) &= x + 4y^2 \\
&\vdots
\end{aligned}
$$



I: 1-order Lie Derivative and Gradient

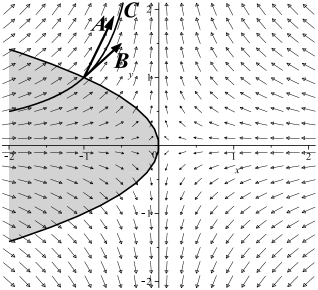
Consider point $(-1,1)$ (see the picture),

- The points on the parabola $p(x,y) = 0$ with zero energy, and the points in the white area have positive energy, i.e. $p(x,y) > 0$.
- B denotes the evolution direction of $\mathbf{f}$ at the point.
- A is the gradient $\nabla p|_{(-1,1)}$ of $p(x,y)$.
- $L_{\mathbf{f}}^{1} p|_{(-1,1)} = 3$ predicts that the trajectory starting at $(-1,1)$ will enter the white area.

Example

Let $\mathbf{f} = (-x, y)$ and $p(x,y) = x + y^2$. Then

$$\begin{aligned} L^0_{\mathbf{f}} p(x,y) &= x + y^2 \\ L^1_{\mathbf{f}} p(x,y) &= -x + 2y^2 \\ L^2_{\mathbf{f}} p(x,y) &= x + 4y^2 \\ &\vdots \end{aligned}$$



I: 1-order Lie Derivative and Gradient

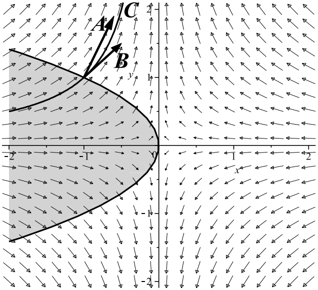
Consider point $(-1, 1)$ (see the picture),

- The points on the parabola $p(x,y) = 0$ with zero energy, and the points in the white area have positive energy, i.e. $p(x,y) > 0$.
- B denotes the evolution direction of $\mathbf{f}$ at the point.
- A is the gradient $\nabla p|_{(-1,1)}$ of $p(x,y)$.
- $L^1_{\mathbf{f}} p|_{(-1,1)} = 3$ predicts that the trajectory starting at $(-1, 1)$ will enter the white area.

Example

Let $\mathbf{f} = (-x, y)$ and $p(x,y) = x + y^2$. Then

$$\begin{aligned} L_{\mathbf{f}}^0 p(x,y) &= x + y^2 \\ L_{\mathbf{f}}^1 p(x,y) &= -x + 2y^2 \\ L_{\mathbf{f}}^2 p(x,y) &= x + 4y^2 \\ &\vdots \end{aligned}$$



I: 1-order Lie Derivative and Gradient

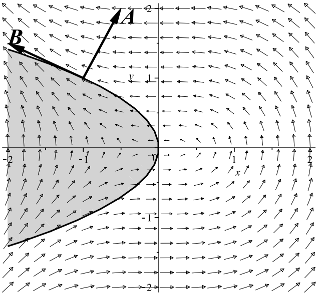
Consider point $(-1,1)$ (see the picture),

- The points on the parabola $p(x,y) = 0$ with zero energy, and the points in the white area have positive energy, i.e. $p(x,y) > 0$.
- B denotes the evolution direction of $\mathbf{f}$ at the point.
- A is the gradient $\nabla p|_{(-1,1)}$ of $p(x,y)$.
- $L_{\mathbf{f}}^1 p|_{(-1,1)} = 3$ predicts that the trajectory starting at $(-1,1)$ will enter the white area.

Example

Let $\mathbf{f}(x, y) = (-2y, x^2)$ and $h(x, y) = x + y^2$. Then

$$
\begin{aligned}
L^0_{\mathbf{f}} h(x,y) &= x + y^2 \\
L^1_{\mathbf{f}} h(x,y) &= -2y + 2x^2 y \\
L^2_{\mathbf{f}} h(x,y) &= -8y^2 x - (2 - 2x^2)x^2 \\
&\vdots
\end{aligned}
$$



II: Demand for Higher Order Lie Derivative

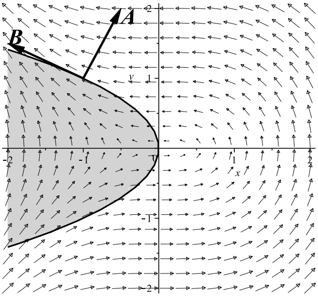
Also consider point $(-1, 1)$ on $h(x, y) = 0$ (see the picture),

- the gradient of h is $(1, 2)$ (vector A);
- the evolution direction is $(-2, 1)$ (vector B);
- their inner product is zero, i.e., $L^1_{\mathbf{f}} h(-1, 1) = 0$, thus it is impossible to predict the tendency of the trajectory starting from $(-1, 1)$ via the 1-order Lie derivative;
- By a simple computation, $L^2_{\mathbf{f}} h(-1, 1) = 8$. Hence $\gamma_{h,\mathbf{f}}(-1, 1) = 2$.

Example

Let $\mathbf{f}(x,y) = (-2y, x^2)$ and $h(x,y) = x + y^2$. Then

$$L^0_{\mathbf{f}} h(x,y) = x + y^2$$
$$L^1_{\mathbf{f}} h(x,y) = -2y + 2x^2 y$$
$$L^2_{\mathbf{f}} h(x,y) = -8y^2 x - (2 - 2x^2)x^2$$
$$\vdots$$



II: Demand for Higher Order Lie Derivative

Also consider point $(-1,1)$ on $h(x,y) = 0$ (see the picture),

- the gradient of h is $(1,2)$ (vector A);
- the evolution direction is $(-2,1)$ (vector B);
- their inner product is zero, i.e., $L^1_{\mathbf{f}} h(-1,1) = 0$, thus it is impossible to predict the tendency of the trajectory starting from $(-1,1)$ via the 1-order Lie derivative;
- By a simple computation, $L^2_{\mathbf{f}} h(-1,1) = 8$. Hence $\gamma_{h,\mathbf{f}}(-1,1) = 2$.

Theoretical Results

Theorem (Rank Theorem)

Given a polynomial p and a PVF $\mathbf{f}$, there is a natural number $N_{p,\mathbf{f}}$ such that for any $\mathbf{x} \in \mathbb{R}^n$, if $\gamma_{p,\mathbf{f}}(\mathbf{x}) < \infty$, then $\gamma_{p,\mathbf{f}}(\mathbf{x}) \leq N_{p,\mathbf{f}}$.

Theorem (Parametric Rank Theorem)

Given a parametric polynomial $p(\mathbf{u}, \mathbf{x})$ and a PVF $\mathbf{f}$, there is an integer $N_{p,\mathbf{f}} \in \mathbb{N}$ such that $\gamma_{p_{\mathbf{u}_0},\mathbf{f}}(\mathbf{x}) < \infty$ implies $\gamma_{p_{\mathbf{u}_0},\mathbf{f}}(\mathbf{x}) \leq N_{p,\mathbf{f}}$ for all $\mathbf{x} \in \mathbb{R}^n$ and all $\mathbf{u}_0 \in \mathbb{R}^w$.

Theorem (Criterion Theorem)

Given a polynomial p, $p(\mathbf{x}) \geq 0$ is an SCI of the PCCDS $(h(\mathbf{x}) \geq 0, \mathbf{f})$ iff

$$\theta(h, p, \mathbf{f}, \mathbf{x}) \widehat{=} (p(\mathbf{x}) = 0 \wedge \pi(p, \mathbf{f}, \mathbf{x})) \rightarrow \pi(h, \mathbf{f}, \mathbf{x}), \tag{3}$$

holds for all $\mathbf{x} \in \mathbb{R}^n$, where

$$\pi^{(i)}(p, \mathbf{f}, \mathbf{x}) \quad \widehat{=} \quad \left(\bigwedge_{0 \leq j < i} L_{\mathbf{f}}^j p(\mathbf{x}) = 0 \right) \wedge L_{\mathbf{f}}^i p(\mathbf{x}) < 0\,,$$

$$\pi(p, \mathbf{f}, \mathbf{x}) \quad \widehat{=} \quad \bigvee_{0 \leq i \leq N_{p,\mathbf{f}}} \pi^{(i)}(p, \mathbf{f}, \mathbf{x})\,.$$

Theoretical Results

Theorem (Rank Theorem)

Given a polynomial p and a PVF $\mathbf{f}$, there is a natural number $N_{p,\mathbf{f}}$ such that for any $\mathbf{x} \in \mathbb{R}^n$, if $\gamma_{p,\mathbf{f}}(\mathbf{x}) < \infty$, then $\gamma_{p,\mathbf{f}}(\mathbf{x}) \leq N_{p,\mathbf{f}}$.

Theorem (Parametric Rank Theorem)

Given a parametric polynomial $p(\mathbf{u},\mathbf{x})$ and a PVF $\mathbf{f}$, there is an integer $N_{p,\mathbf{f}} \in \mathbb{N}$ such that $\gamma_{p_{\mathbf{u_0}},\mathbf{f}}(\mathbf{x}) < \infty$ implies $\gamma_{p_{\mathbf{u_0}},\mathbf{f}}(\mathbf{x}) \leq N_{p,\mathbf{f}}$ for all $\mathbf{x} \in \mathbb{R}^n$ and all $\mathbf{u}_0 \in \mathbb{R}^w$.

Theorem (Criterion Theorem)

Given a polynomial p, $p(\mathbf{x}) \geq 0$ is an SCI of the PCCDS $(h(\mathbf{x}) \geq 0, \mathbf{f})$ iff

$$\theta(h,p,\mathbf{f},\mathbf{x}) \widehat{=} (p(\mathbf{x}) = 0 \wedge \pi(p,\mathbf{f},\mathbf{x})) \rightarrow \pi(h,\mathbf{f},\mathbf{x}), \tag{3}$$

holds for all $\mathbf{x} \in \mathbb{R}^n$, where

$$\pi^{(i)}(p,\mathbf{f},\mathbf{x}) \ \widehat{=}\ \left(\bigwedge_{0\leq j<i} L_{\mathbf{f}}^{j} p(\mathbf{x}) = 0\right) \wedge L_{\mathbf{f}}^{i} p(\mathbf{x}) < 0\,,$$

$$\pi(p,\mathbf{f},\mathbf{x}) \ \widehat{=}\ \bigvee_{0\leq i\leq N_{p,\mathbf{f}}} \pi^{(i)}(p,\mathbf{f},\mathbf{x})\,.$$

Theoretical Results

Theorem (Rank Theorem)

Given a polynomial p and a PVF $\mathbf{f}$, there is a natural number $N_{p,\mathbf{f}}$ such that for any $\mathbf{x} \in \mathbb{R}^n$, if $\gamma_{p,\mathbf{f}}(\mathbf{x}) < \infty$, then $\gamma_{p,\mathbf{f}}(\mathbf{x}) \leq N_{p,\mathbf{f}}$.

Theorem (Parametric Rank Theorem)

Given a parametric polynomial $p(\mathbf{u}, \mathbf{x})$ and a PVF $\mathbf{f}$, there is an integer $N_{p,\mathbf{f}} \in \mathbb{N}$ such that $\gamma_{p_{\mathbf{u_0}},\mathbf{f}}(\mathbf{x}) < \infty$ implies $\gamma_{p_{\mathbf{u_0}},\mathbf{f}}(\mathbf{x}) \leq N_{p,\mathbf{f}}$ for all $\mathbf{x} \in \mathbb{R}^n$ and all $\mathbf{u}_0 \in \mathbb{R}^w$.

Theorem (Criterion Theorem)

Given a polynomial p, $p(\mathbf{x}) \geq 0$ is an SCI of the PCCDS $(h(\mathbf{x}) \geq 0, \mathbf{f})$ iff

$$\theta(h, p, \mathbf{f}, \mathbf{x}) \widehat{=} (p(\mathbf{x}) = 0 \wedge \pi(p, \mathbf{f}, \mathbf{x})) \rightarrow \pi(h, \mathbf{f}, \mathbf{x}), \tag{3}$$

holds for all $\mathbf{x} \in \mathbb{R}^n$, where

$$\pi^{(i)}(p, \mathbf{f}, \mathbf{x}) \ \widehat{=} \ \left(\bigwedge_{0 \leq j < i} L^j_{\mathbf{f}} p(\mathbf{x}) = 0 \right) \wedge L^i_{\mathbf{f}} p(\mathbf{x}) < 0\,,$$

$$\pi(p, \mathbf{f}, \mathbf{x}) \ \widehat{=} \ \bigvee_{0 \leq i \leq N_{p,\mathbf{f}}} \pi^{(i)}(p, \mathbf{f}, \mathbf{x})\,.$$

Algorithm

I. **First, set a simple semi-algebraic template $P \mathrel{\widehat{=}} p(\mathbf{u},\mathbf{x}) \geq 0$ using a parametric polynomial $p(\mathbf{u},\mathbf{x})$.**

II. Then apply QE to the formula $\forall \mathbf{x}.\theta(h,p,\mathbf{f},\mathbf{x})$. In practice, QE may be applied to a formula $\forall \mathbf{x}.(\theta \wedge \phi)$, where ϕ is a formula imposing some additional constraint on the SCI P. If the output of QE is *false*, then there is no SCI in the form of the predefined P; otherwise, a constraint on $\mathbf{u}$, denoted by $R(\mathbf{u})$, will be returned.

III. Now, use an SMT solver like Z3 to pick a $\mathbf{u}_0 \in R(\mathbf{u})$ and then $p_{\mathbf{u}_0}(\mathbf{x}) \geq 0$ is an SCI of $(h(\mathbf{x}) \geq 0, \mathbf{f})$.

Algorithm

I. First, set a simple semi-algebraic template $P \,\widehat{=}\, p(\mathbf{u}, \mathbf{x}) \geq 0$ using a parametric polynomial $p(\mathbf{u}, \mathbf{x})$.

II. Then apply **QE** to the formula $\forall \mathbf{x}.\theta(h, p, \mathbf{f}, \mathbf{x})$. In practice, **QE** may be applied to a formula $\forall \mathbf{x}.(\theta \wedge \phi)$, where ϕ is a formula imposing some additional constraint on the SCI P. If the output of **QE** is *false*, then there is no SCI in the form of the predefined P; otherwise, a constraint on $\mathbf{u}$, denoted by $R(\mathbf{u})$, will be returned.

III. Now, use an SMT solver like **Z3** to pick a $\mathbf{u}_0 \in R(\mathbf{u})$ and then $p_{\mathbf{u}_0}(\mathbf{x}) \geq 0$ is an SCI of $(h(\mathbf{x}) \geq 0, \mathbf{f})$.

Algorithm

I. First, set a simple semi-algebraic template $P \mathrel{\widehat{=}} p(\mathbf{u}, \mathbf{x}) \geq 0$ using a parametric polynomial $p(\mathbf{u}, \mathbf{x})$.

II. Then apply **QE** to the formula $\forall \mathbf{x}.\theta(h, p, \mathbf{f}, \mathbf{x})$. In practice, **QE** may be applied to a formula $\forall \mathbf{x}.(\theta \wedge \phi)$, where ϕ is a formula imposing some additional constraint on the SCI P. If the output of **QE** is *false*, then there is no SCI in the form of the predefined P; otherwise, a constraint on $\mathbf{u}$, denoted by $R(\mathbf{u})$, will be returned.

III. Now, use an SMT solver like **Z3** to pick a $\mathbf{u}_0 \in R(\mathbf{u})$ and then $p_{\mathbf{u}_0}(\mathbf{x}) \geq 0$ is an SCI of $(h(\mathbf{x}) \geq 0, \mathbf{f})$.

Running Example

Consider a PDS $(D = -x - y^2 \geq 0, \mathbf{f}(x, y) = (-2y, x^2))$.
Apply procedure (I-III), we have:

I Set a template $P \mathrel{\widehat{=}} p(\mathbf{u}, \mathbf{x}) \geq 0$ with $p(\mathbf{u}, \mathbf{x}) \mathrel{\widehat{=}} ay(x - y)$, where $\mathbf{u} \mathrel{\widehat{=}} (a)$. By a simple computation we get $N_{p,\mathbf{f}} = 2$.

II Compute the corresponding formula

$$\begin{aligned} \theta(h, p, \mathbf{f}, \mathbf{x}) \quad &\widehat{=} \quad p = 0 \wedge (\pi^{(0)}_{p,\mathbf{f},\mathbf{x}} \vee \pi^{(1)}_{p,\mathbf{f},\mathbf{x}} \vee \pi^{(2)}_{p,\mathbf{f},\mathbf{x}}) \longrightarrow \\ & \qquad (\pi^{(0)}_{h,\mathbf{f},\mathbf{x}} \vee \pi^{(1)}_{h,\mathbf{f},\mathbf{x}} \vee \pi^{(2)}_{h,\mathbf{f},\mathbf{x}}) \end{aligned}$$

where

$$\begin{aligned} \pi^{(0)}_{h,\mathbf{f},\mathbf{x}} \quad &\widehat{=} \quad -x - y^2 < 0, \\ \pi^{(1)}_{h,\mathbf{f},\mathbf{x}} \quad &\widehat{=} \quad -x - y^2 = 0 \wedge 2y - 2x^2y < 0, \\ \pi^{(2)}_{h,\mathbf{f},\mathbf{x}} \quad &\widehat{=} \quad -x - y^2 = 0 \wedge 2y - 2x^2y = 0 \wedge 8xy^2 + 2x^2 - 2x^4 < 0, \\ \pi^{(0)}_{p,\mathbf{f},\mathbf{x}} \quad &\widehat{=} \quad ay(x - y) < 0, \\ \pi^{(1)}_{p,\mathbf{f},\mathbf{x}} \quad &\widehat{=} \quad ay(x - y) = 0 \wedge -2ay^2 + ax^3 - 2yax^2 < 0, \\ \pi^{(2)}_{p,\mathbf{f},\mathbf{x}} \quad &\widehat{=} \quad ay(x - y) = 0 \wedge -2ay^2 + ax^3 - 2yax^2 = 0 \\ & \qquad \wedge 40axy^2 - 16ay^3 + 32ax^3y - 10ax^4 < 0. \end{aligned}$$

Running Example

Consider a PDS $(D = -x - y^2 \geq 0, \mathbf{f}(x, y) = (-2y, x^2))$.
Apply procedure (I-III), we have:

I Set a template $P \,\widehat{=}\, p(\mathbf{u}, \mathbf{x}) \geq 0$ with $p(\mathbf{u}, \mathbf{x}) \,\widehat{=}\, ay(x - y)$, where $\mathbf{u} \,\widehat{=}\, (a)$. By a simple computation we get $N_{p,\mathbf{f}} = 2$.

II Compute the corresponding formula

$$\begin{aligned}\theta(h, p, \mathbf{f}, \mathbf{x}) \quad \widehat{=} \quad & p = 0 \wedge (\pi^{(0)}_{p,\mathbf{f},\mathbf{x}} \vee \pi^{(1)}_{p,\mathbf{f},\mathbf{x}} \vee \pi^{(2)}_{p,\mathbf{f},\mathbf{x}}) \longrightarrow \\ & (\pi^{(0)}_{h,\mathbf{f},\mathbf{x}} \vee \pi^{(1)}_{h,\mathbf{f},\mathbf{x}} \vee \pi^{(2)}_{h,\mathbf{f},\mathbf{x}})\end{aligned}$$

where

$$\begin{aligned}
\pi^{(0)}_{h,\mathbf{f},\mathbf{x}} \quad & \widehat{=} \quad -x - y^2 < 0, \\
\pi^{(1)}_{h,\mathbf{f},\mathbf{x}} \quad & \widehat{=} \quad -x - y^2 = 0 \wedge 2y - 2x^2y < 0, \\
\pi^{(2)}_{h,\mathbf{f},\mathbf{x}} \quad & \widehat{=} \quad -x - y^2 = 0 \wedge 2y - 2x^2y = 0 \wedge 8xy^2 + 2x^2 - 2x^4 < 0, \\
\pi^{(0)}_{p,\mathbf{f},\mathbf{x}} \quad & \widehat{=} \quad ay(x - y) < 0, \\
\pi^{(1)}_{p,\mathbf{f},\mathbf{x}} \quad & \widehat{=} \quad ay(x - y) = 0 \wedge -2ay^2 + ax^3 - 2yax^2 < 0, \\
\pi^{(2)}_{p,\mathbf{f},\mathbf{x}} \quad & \widehat{=} \quad ay(x - y) = 0 \wedge -2ay^2 + ax^3 - 2yax^2 = 0 \\
& \qquad \wedge 40axy^2 - 16ay^3 + 32ax^3y - 10ax^4 < 0.
\end{aligned}$$

Running Example

Consider a PDS $(D = -x - y^2 \geq 0, \mathbf{f}(x,y) = (-2y, x^2))$.

Apply procedure (I-III), we have:

I Set a template $P \mathrel{\widehat{=}} p(\mathbf{u}, \mathbf{x}) \geq 0$ with $p(\mathbf{u}, \mathbf{x}) \mathrel{\widehat{=}} ay(x-y)$, where $\mathbf{u} \mathrel{\widehat{=}} (a)$. By a simple computation we get $N_{p,\mathbf{f}} = 2$.

II Compute the corresponding formula

$$\begin{array}{rcl} \theta(h, p, \mathbf{f}, \mathbf{x}) & \widehat{=} & p = 0 \wedge (\pi^{(0)}_{p,\mathbf{f},\mathbf{x}} \vee \pi^{(1)}_{p,\mathbf{f},\mathbf{x}} \vee \pi^{(2)}_{p,\mathbf{f},\mathbf{x}}) \longrightarrow \\ & & (\pi^{(0)}_{h,\mathbf{f},\mathbf{x}} \vee \pi^{(1)}_{h,\mathbf{f},\mathbf{x}} \vee \pi^{(2)}_{h,\mathbf{f},\mathbf{x}}) \end{array}$$

where

$$\begin{array}{rcl} \pi^{(0)}_{h,\mathbf{f},\mathbf{x}} & \widehat{=} & -x - y^2 < 0, \\ \pi^{(1)}_{h,\mathbf{f},\mathbf{x}} & \widehat{=} & -x - y^2 = 0 \wedge 2y - 2x^2y < 0, \\ \pi^{(2)}_{h,\mathbf{f},\mathbf{x}} & \widehat{=} & -x - y^2 = 0 \wedge 2y - 2x^2y = 0 \wedge 8xy^2 + 2x^2 - 2x^4 < 0, \\ \pi^{(0)}_{p,\mathbf{f},\mathbf{x}} & \widehat{=} & ay(x-y) < 0, \\ \pi^{(1)}_{p,\mathbf{f},\mathbf{x}} & \widehat{=} & ay(x-y) = 0 \wedge -2ay^2 + ax^3 - 2yax^2 < 0, \\ \pi^{(2)}_{p,\mathbf{f},\mathbf{x}} & \widehat{=} & ay(x-y) = 0 \wedge -2ay^2 + ax^3 - 2yax^2 = 0 \\ & & \wedge\, 40axy^2 - 16ay^3 + 32ax^3y - 10ax^4 < 0. \end{array}$$

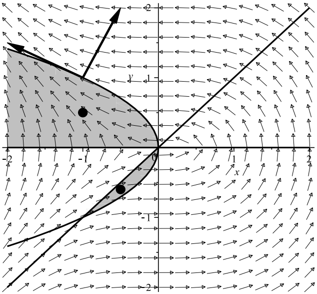
Running Example (Cont'd)

III In addition, we require the two points $\{(-1, 0.5), (-0.5, -0.6)\}$ to be contained in P. Then apply **QE** to the formula

$$\forall x \forall y. (\theta(h, p, \mathbf{f}, \mathbf{x}) \wedge 0.5a(-1-0.5) \geq 0 \wedge -0.6a(-0.5+0.6) \geq 0).$$

The result is $a \leq 0$.

IV Just pick $a = -1$, and then $-xy + y^2 \geq 0$ is an SCI of $(D, \mathbf{f})$. The grey part of Picture III is the intersection of the invariant P and domain D.



III: SCI in Simple Case

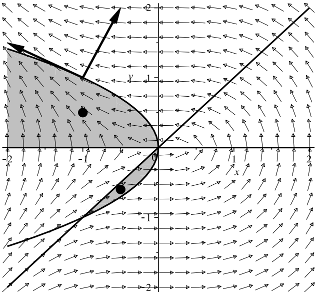
Running Example (Cont'd)

III In addition, we require the two points $\{(-1, 0.5), (-0.5, -0.6)\}$ to be contained in P. Then apply **QE** to the formula

$$\forall x \forall y. (\theta(h, p, \mathbf{f}, \mathbf{x}) \wedge 0.5a(-1 - 0.5) \geq 0 \wedge -0.6a(-0.5 + 0.6) \geq 0).$$

The result is $a \leq 0$.

IV Just pick $a = -1$, and then $-xy + y^2 \geq 0$ is an SCI of $(D, \mathbf{f})$. The grey part of Picture III is the intersection of the invariant P and domain D.



III: SCI in Simple Case

Outline

General Case

- **Problem:** Consider a PDS $(D, \mathbf{f})$ with

$$D = \bigvee_{i=1}^{I} \bigwedge_{j=1}^{J_i} p_{ij}(\mathbf{x}) \triangleright 0,$$

and $\mathbf{f} \in \mathbb{Q}^n[\mathbf{x}]$, where $\triangleright \in \{\geq, >\}$, to generate SAIs automatically with a general template

$$P = \bigvee_{k=1}^{K} \bigwedge_{l=1}^{L_k} p_{kl}(\mathbf{u}_{kl}, \mathbf{x}) \triangleright 0\,,\ \triangleright \in \{\geq, >\}$$

- **Basic idea** The procedure is essentially same as in the simple case, but have to sophisticatedly handle the complex combinations due to the complicated boundaries.

General Case

- **Problem:** Consider a PDS $(D, \mathbf{f})$ with

$$D = \bigvee_{i=1}^{I} \bigwedge_{j=1}^{J_i} p_{ij}(\mathbf{x}) \triangleright 0,$$

and $\mathbf{f} \in \mathbb{Q}^n[\mathbf{x}]$, where $\triangleright \in \{\geq, >\}$, to generate SAIs automatically with a general template

$$P = \bigvee_{k=1}^{K} \bigwedge_{l=1}^{L_k} p_{kl}(\mathbf{u}_{kl}, \mathbf{x}) \triangleright 0\,, \ \triangleright \in \{\geq, >\}$$

- **Basic idea** The procedure is essentially same as in the simple case, but have to sophisticatedly handle the complex combinations due to the complicated boundaries.

Theorem (Main Result)

A semi-algebraic template $P(\mathbf{u},\mathbf{x})$ *defined by*

$$\bigvee_{k=1}^{K}\left(\bigwedge_{j=1}^{j_k} p_{kj}(\mathbf{u}_{kj},\mathbf{x}) \geq 0 \quad \wedge \bigwedge_{j=j_k+1}^{J_k} p_{kj}(\mathbf{u}_{kj},\mathbf{x}) > 0\right)$$

is a CI of the PCCDS $(D,\mathbf{f})$ *with*

$$D \mathrel{\widehat{=}} \bigvee_{m=1}^{M}\left(\bigwedge_{l=1}^{l_m} p_{ml}(\mathbf{x}) \geq 0 \quad \wedge \bigwedge_{l=l_m+1}^{L_m} p_{ml}(\mathbf{x}) > 0\right),$$

iff $\mathbf{u}$ *satisfies*

$$\forall \mathbf{x}.\Big(\big(P \wedge D \wedge \Phi_D \rightarrow \Phi_P\big) \wedge \big(\neg P \wedge D \wedge \Phi_D^{Iv} \rightarrow \neg\Phi_P^{Iv}\big)\Big),$$

where

Theorem (Main Result (Cont'd))

$$\Phi_D \widehat{=} \bigvee_{m=1}^{M} \left(\bigwedge_{l=1}^{l_m} \psi_0^+(p_{ml}, \mathbf{f}) \wedge \bigwedge_{l=l_m+1}^{L_m} \psi^+(p_{ml}, \mathbf{f}) \right),$$

$$\Phi_P \widehat{=} \bigvee_{k=1}^{K} \left(\bigwedge_{j=1}^{j_k} \psi_0^+(p_{kj}, \mathbf{f}) \wedge \bigwedge_{j=j_k+1}^{J_k} \psi^+(p_{kj}, \mathbf{f}) \right),$$

$$\Phi_D^{Iv} \widehat{=} \bigvee_{m=1}^{M} \left(\bigwedge_{l=1}^{l_m} \varphi_0^+(p_{ml}, \mathbf{f}) \wedge \bigwedge_{l=l_m+1}^{L_m} \varphi^+(p_{ml}, \mathbf{f}) \right),$$

$$\Phi_P^{Iv} \widehat{=} \bigvee_{k=1}^{K} \left(\bigwedge_{j=1}^{j_k} \varphi_0^+(p_{kj}, \mathbf{f}) \wedge \bigwedge_{j=j_k+1}^{J_k} \varphi^+(p_{kj}, \mathbf{f}) \right),$$

$$\psi^+(p, \mathbf{f}) \widehat{=} \bigvee_{0 \le i \le N_{p,\mathbf{f}}} \psi^{(i)}(p, \mathbf{f}) \text{ with } \psi^{(i)}(p, \mathbf{f}) \widehat{=} \left(\bigwedge_{0 \le j < i} L_{\mathbf{f}}^j p = 0 \right) \wedge L_{\mathbf{f}}^i p > 0, \text{ and}$$

$$\psi_0^+(p, \mathbf{f}) \widehat{=} \psi^+(p, \mathbf{f}) \vee \left(\bigwedge_{0 \le j \le N_{p,\mathbf{f}}} L_{\mathbf{f}}^j p = 0 \right)$$

$$\varphi^+(p, \mathbf{f}) \widehat{=} \bigvee_{0 \le i \le N_{p,\mathbf{f}}} \varphi^{(i)}(p, \mathbf{f}) \text{ with } \varphi^{(i)}(p, \mathbf{f}) \widehat{=} \left(\bigwedge_{0 \le j < i} L_{\mathbf{f}}^j p = 0 \right) \wedge (-1)^i \cdot L_{\mathbf{f}}^i p > 0, \text{ and}$$

$$\varphi_0^+(p, \mathbf{f}) \widehat{=} \varphi^+(p, \mathbf{f}) \vee \left(\bigwedge_{0 \le j \le N_{p,\mathbf{f}}} L_{\mathbf{f}}^j p = 0 \right).$$

Running Example

- Let $\mathbf{f}(x, y) = (-2y, x^2)$ and $D \mathrel{\widehat{=}} \mathbb{R}^2$.
- Take a template: $P(\mathbf{u}, \mathbf{x}) \mathrel{\widehat{=}} x - a \geq 0 \vee y - b > 0$ with $\mathbf{u} = (a, b)$.
- So, P is an SCI of $(D, \mathbf{f})$ iff a, b satisfy

$$\forall x \forall y.(P \rightarrow \zeta) \wedge (\neg P \rightarrow \neg \xi),$$

where

$$\begin{aligned}
\zeta \mathrel{\widehat{=}} & (x - a > 0) \vee (x - a = 0 \wedge -2y > 0) \\
& \vee (x - a = 0 \wedge -2y = 0 \wedge -2x^2 \geq 0) \\
& \vee (y - b > 0) \vee (y - b = 0 \wedge x^2 > 0) \\
& \vee (y - b = 0 \wedge x^2 = 0 \wedge -4yx > 0) \\
& \vee (y - b = 0 \wedge x^2 = 0 \wedge -4yx = 0 \wedge 8y^2 - 4x^3 > 0) \\
\xi \mathrel{\widehat{=}} & (x - a > 0) \vee (x - a = 0 \wedge -2y < 0) \\
& \vee (x - a = 0 \wedge -2y = 0 \wedge -2x^2 \geq 0) \\
& \vee (y - b > 0) \vee (y - b = 0 \wedge x^2 < 0) \\
& \vee (y - b = 0 \wedge x^2 = 0 \wedge -4yx > 0) \\
& \vee (y - b = 0 \wedge x^2 = 0 \wedge -4yx = 0 \wedge 8y^2 - 4x^3 < 0)
\end{aligned}$$

Running Example

- Let $\mathbf{f}(x,y) = (-2y, x^2)$ and $D \mathrel{\widehat{=}} \mathbb{R}^2$.
- Take a template: $P(\mathbf{u},\mathbf{x}) \mathrel{\widehat{=}} x - a \geq 0 \vee y - b > 0$ with $\mathbf{u} = (a, b)$.
- So, P is an SCI of $(D, \mathbf{f})$ iff a, b satisfy

$$\forall x \forall y.(P \to \zeta) \wedge (\neg P \to \neg \xi),$$

where

$$\begin{aligned}
\zeta \mathrel{\widehat{=}} & (x-a>0) \vee (x-a=0 \wedge -2y>0) \\
& \vee (x-a=0 \wedge -2y=0 \wedge -2x^2 \geq 0) \\
& \vee (y-b>0) \vee (y-b=0 \wedge x^2>0) \\
& \vee (y-b=0 \wedge x^2=0 \wedge -4yx>0) \\
& \vee (y-b=0 \wedge x^2=0 \wedge -4yx=0 \wedge 8y^2-4x^3>0) \\
\xi \mathrel{\widehat{=}} & (x-a>0) \vee (x-a=0 \wedge -2y<0) \\
& \vee (x-a=0 \wedge -2y=0 \wedge -2x^2 \geq 0) \\
& \vee (y-b>0) \vee (y-b=0 \wedge x^2<0) \\
& \vee (y-b=0 \wedge x^2=0 \wedge -4yx>0) \\
& \vee (y-b=0 \wedge x^2=0 \wedge -4yx=0 \wedge 8y^2-4x^3<0)
\end{aligned}$$

Running Example

- Let $\mathbf{f}(x, y) = (-2y, x^2)$ and $D \mathrel{\widehat{=}} \mathbb{R}^2$.
- Take a template: $P(\mathbf{u}, \mathbf{x}) \mathrel{\widehat{=}} x - a \geq 0 \vee y - b > 0$ with $\mathbf{u} = (a, b)$.
- So, P is an SCI of $(D, \mathbf{f})$ iff a, b satisfy

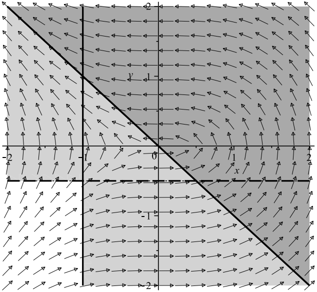
$$\forall x \forall y.(P \rightarrow \zeta) \wedge (\neg P \rightarrow \neg \xi),$$

where

$$\begin{aligned}
\zeta \mathrel{\widehat{=}} & (x - a > 0) \vee (x - a = 0 \wedge -2y > 0) \\
& \vee (x - a = 0 \wedge -2y = 0 \wedge -2x^2 \geq 0) \\
& \vee (y - b > 0) \vee (y - b = 0 \wedge x^2 > 0) \\
& \vee (y - b = 0 \wedge x^2 = 0 \wedge -4yx > 0) \\
& \vee (y - b = 0 \wedge x^2 = 0 \wedge -4yx = 0 \wedge 8y^2 - 4x^3 > 0) \\
\xi \mathrel{\widehat{=}} & (x - a > 0) \vee (x - a = 0 \wedge -2y < 0) \\
& \vee (x - a = 0 \wedge -2y = 0 \wedge -2x^2 \geq 0) \\
& \vee (y - b > 0) \vee (y - b = 0 \wedge x^2 < 0) \\
& \vee (y - b = 0 \wedge x^2 = 0 \wedge -4yx > 0) \\
& \vee (y - b = 0 \wedge x^2 = 0 \wedge -4yx = 0 \wedge 8y^2 - 4x^3 < 0)
\end{aligned}$$

Running Example (Cont'd)

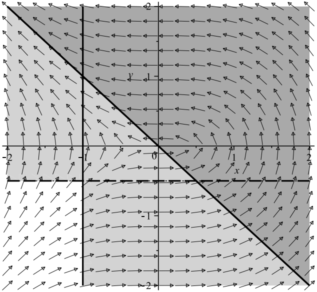
- In addition, we require the set $x + y \geq 0$ to be contained in P.
- By applying QE, we get $a + b \leq 0 \wedge b \leq 0$.
- Let $a = -1$ and $b = -0.5$, and we obtain an SCI $P \hat{=} x + 1 \geq 0 \vee y + 0.5 > 0$.



IV: SCI in General Case

Running Example (Cont'd)

- In addition, we require the set $x + y \geq 0$ to be contained in P.
- By applying **QE**, we get $a + b \leq 0 \wedge b \leq 0$.
- Let $a = -1$ and $b = -0.5$, and we obtain an SCI $P \,\widehat{=}\, x + 1 \geq 0 \vee y + 0.5 > 0$.



IV: SCI in General Case

Outline

Algorithm

I. Predefine a familiy of semi-algebraic templates $I_q(\mathbf{u}, \mathbf{x})$ with degree bound d for each $q \in Q$, as the SCI to be generated at mode q.

II. Translate conditions for the family of $I_q(\mathbf{u}, \mathbf{x})$ to be a GI of $\mathcal{H}$, i.e.

- $\Xi_q \subseteq I_q$ for all $q \in Q$;
- for any $e = (q, q') \in E$, if $\mathbf{x} \in I_q \cap G_e$, then $\mathbf{x}' = R_e(\mathbf{x}) \in I_{q'}$;
- for any $q \in Q$, I_q is a CI of $(D_q, \mathbf{f}_q)$

into a set of first-order real arithmetic formulas, i.e.

(1) $\forall \mathbf{x}. (\Xi_q \rightarrow I_q(\mathbf{u}, \mathbf{x}))$ for all $q \in Q$;

(2) $\forall \mathbf{x}, \mathbf{x}'. (I_q(\mathbf{u}, \mathbf{x}) \wedge G_e \wedge \mathbf{x}' = R_e(\mathbf{x}) \rightarrow I_{q'}(\mathbf{u}, \mathbf{x}'))$ for all $q \in Q$ and all $e = (q, q') \in E$;

(3) $\forall \mathbf{x}. ((I_q(\mathbf{u}, \mathbf{x}) \wedge D_q \wedge \Phi_{D_q} \rightarrow \Phi_{I_q}) \wedge (\neg I_q(\mathbf{u}, \mathbf{x}) \wedge D_q \wedge \Phi^{\mathrm{Iv}}_{D_q} \rightarrow \neg \Phi^{\mathrm{Iv}}_{I_q}))$, for each $q \in Q$.

For safety property $\mathcal{S}$, there may be a fourth set of formulas:

(4) $\forall \mathbf{x}. (I_q(\mathbf{u}, \mathbf{x}) \longrightarrow S_q)$ for all $q \in Q$.

III. Take the conjunction of all the formulas in Step 2 and apply QE to get a QFF $\phi(\mathbf{u})$. Then choose a specific $\mathbf{u}_0$ from $\phi(\mathbf{u})$ with a tool like **Z3**, and the set of instantiations $I_{q,\mathbf{u}_0}(\mathbf{x})$ form a GI of $\mathcal{H}$.

Algorithm

I. Predefine a familiy of semi-algebraic templates $I_q(\mathbf{u}, \mathbf{x})$ with degree bound d for each $q \in Q$, as the SCI to be generated at mode q.

II. Translate conditions for the family of $I_q(\mathbf{u}, \mathbf{x})$ to be a GI of $\mathcal{H}$, i.e.

- $\Xi_q \subseteq I_q$ for all $q \in Q$;
- for any $e = (q, q') \in E$, if $\mathbf{x} \in I_q \cap G_e$, then $\mathbf{x}' = R_e(\mathbf{x}) \in I_{q'}$;
- for any $q \in Q$, I_q is a CI of $(D_q, \mathbf{f}_q)$

into a set of first-order real arithmetic formulas, i.e.

(1) $\forall \mathbf{x}. \big(\Xi_q \to I_q(\mathbf{u}, \mathbf{x})\big)$ for all $q \in Q$;

(2) $\forall \mathbf{x}, \mathbf{x}'. \big(I_q(\mathbf{u}, \mathbf{x}) \wedge G_e \wedge \mathbf{x}' = R_e(\mathbf{x}) \to I_{q'}(\mathbf{u}, \mathbf{x}')\big)$ for all $q \in Q$ and all $e = (q, q') \in E$;

(3) $\forall \mathbf{x}. \big((I_q(\mathbf{u}, \mathbf{x}) \wedge D_q \wedge \Phi_{D_q} \to \Phi_{I_q}) \wedge (\neg I_q(\mathbf{u}, \mathbf{x}) \wedge D_q \wedge \Phi^{\mathrm{Iv}}_{D_q} \to \neg \Phi^{\mathrm{Iv}}_{I_q})\big)$, for each $q \in Q$.

For safety property $\mathcal{S}$, there may be a fourth set of formulas:

(4) $\forall \mathbf{x}. (I_q(\mathbf{u}, \mathbf{x}) \longrightarrow S_q)$ for all $q \in Q$.

III. Take the conjunction of all the formulas in Step 2 and apply QE to get a QFF $\phi(\mathbf{u})$. Then choose a specific $\mathbf{u}_0$ from $\phi(\mathbf{u})$ with a tool like **Z3**, and the set of instantiations $I_{q,\mathbf{u}_0}(\mathbf{x})$ form a GI of $\mathcal{H}$.

Algorithm

I. Predefine a familiy of semi-algebraic templates $I_q(\mathbf{u},\mathbf{x})$ with degree bound d for each $q \in Q$, as the SCI to be generated at mode q.

II. Translate conditions for the family of $I_q(\mathbf{u},\mathbf{x})$ to be a GI of $\mathcal{H}$, i.e.
   - $\Xi_q \subseteq I_q$ for all $q \in Q$;
   - for any $e = (q, q') \in E$, if $\mathbf{x} \in I_q \cap G_e$, then $\mathbf{x}' = R_e(\mathbf{x}) \in I_{q'}$;
   - for any $q \in Q$, I_q is a CI of $(D_q, \mathbf{f}_q)$

   into a set of first-order real arithmetic formulas, i.e.

   (1) $\forall \mathbf{x}. \big(\Xi_q \rightarrow I_q(\mathbf{u},\mathbf{x})\big)$ for all $q \in Q$;

   (2) $\forall \mathbf{x},\mathbf{x}'. \big(I_q(\mathbf{u},\mathbf{x}) \wedge G_e \wedge \mathbf{x}' = R_e(\mathbf{x}) \rightarrow I_{q'}(\mathbf{u},\mathbf{x}')\big)$ for all $q \in Q$ and all $e = (q, q') \in E$;

   (3) $\forall \mathbf{x}. \big((I_q(\mathbf{u},\mathbf{x}) \wedge D_q \wedge \Phi_{D_q} \rightarrow \Phi_{I_q}) \wedge (\neg I_q(\mathbf{u},\mathbf{x}) \wedge D_q \wedge \Phi^{\mathrm{Iv}}_{D_q} \rightarrow \neg \Phi^{\mathrm{Iv}}_{I_q})\big)$, for each $q \in Q$.

   For safety property $\mathcal{S}$, there may be a fourth set of formulas:

   (4) $\forall \mathbf{x}. (I_q(\mathbf{u},\mathbf{x}) \longrightarrow S_q)$ for all $q \in Q$.

III. Take the conjunction of all the formulas in Step 2 and apply QE to get a QFF $\phi(\mathbf{u})$. Then choose a specific $\mathbf{u}_0$ from $\phi(\mathbf{u})$ with a tool like **Z3**, and the set of instantiations $I_{q,\mathbf{u}_0}(\mathbf{x})$ form a GI of $\mathcal{H}$.

Algorithm

I. Predefine a familiy of semi-algebraic templates $I_q(\mathbf{u}, \mathbf{x})$ with degree bound d for each $q \in Q$, as the SCI to be generated at mode q.

II. Translate conditions for the family of $I_q(\mathbf{u}, \mathbf{x})$ to be a GI of $\mathcal{H}$, i.e.

- $\Xi_q \subseteq I_q$ for all $q \in Q$;
- for any $e = (q, q') \in E$, if $\mathbf{x} \in I_q \cap G_e$, then $\mathbf{x}' = R_e(\mathbf{x}) \in I_{q'}$;
- for any $q \in Q$, I_q is a CI of $(D_q, \mathbf{f}_q)$

into a set of first-order real arithmetic formulas, i.e.

(1) $\forall \mathbf{x}. \big(\Xi_q \rightarrow I_q(\mathbf{u}, \mathbf{x})\big)$ for all $q \in Q$;

(2) $\forall \mathbf{x}, \mathbf{x}'. \big(I_q(\mathbf{u}, \mathbf{x}) \wedge G_e \wedge \mathbf{x}' = R_e(\mathbf{x}) \rightarrow I_{q'}(\mathbf{u}, \mathbf{x}')\big)$ for all $q \in Q$ and all $e = (q, q') \in E$;

(3) $\forall \mathbf{x}. \big((I_q(\mathbf{u}, \mathbf{x}) \wedge D_q \wedge \Phi_{D_\mathbf{q}} \rightarrow \Phi_{I_\mathbf{q}}) \wedge (\neg I_q(\mathbf{u}, \mathbf{x}) \wedge D_q \wedge \Phi^{\mathrm{Iv}}_{D_\mathbf{q}} \rightarrow \neg \Phi^{\mathrm{Iv}}_{I_\mathbf{q}})\big)$, for each $q \in Q$.

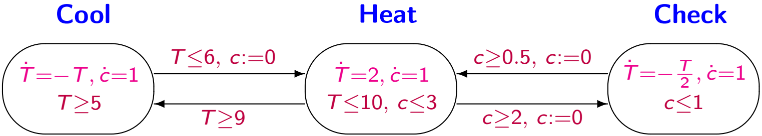
For safety property $\mathcal{S}$, there may be a fourth set of formulas:

(4) $\forall \mathbf{x}. (I_q(\mathbf{u}, \mathbf{x}) \longrightarrow S_q)$ for all $q \in Q$.

III. Take the conjunction of all the formulas in Step 2 and apply QE to get a QFF $\phi(\mathbf{u})$. Then choose a specific $\mathbf{u}_0$ from $\phi(\mathbf{u})$ with a tool like **Z3**, and the set of instantiations $I_{q,\mathbf{u_0}}(\mathbf{x})$ form a GI of $\mathcal{H}$.

Running Example

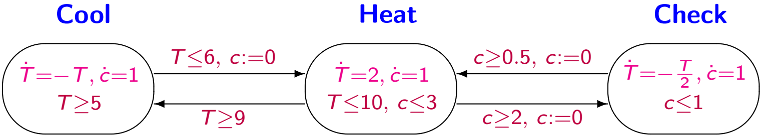
- The Thermostat can be described by the HA in following figure.



- To verify that under the initial condition $\Xi_{\mathcal{H}} \mathrel{\widehat{=}} \{q_{\mathrm{ht}}\} \times X_0$ with $X_0 \mathrel{\widehat{=}} c = 0 \wedge 5 \leq T \leq 10$, $S \mathrel{\widehat{=}} T \geq 4.5$ is satisfied at all modes.

Running Example

- The Thermostat can be described by the HA in following figure.



- To verify that under the initial condition $\Xi_{\mathcal{H}} \mathrel{\widehat{=}} \{q_{\mathrm{ht}}\} \times X_0$ with $X_0 \mathrel{\widehat{=}} c = 0 \wedge 5 \leq T \leq 10$, $S \mathrel{\widehat{=}} T \geq 4.5$ is satisfied at all modes.

Running Example (Cont'd)

- Firstly, predefine the following set of templates:
  - $I_{q_{\mathrm{ht}}} \mathrel{\widehat{=}} T + a_1 c + a_0 \geq 0 \wedge c \geq 0$;
  - $I_{q_{\mathrm{cl}}} \mathrel{\widehat{=}} T + a_2 \geq 0$;
  - $I_{q_{\mathrm{ck}}} \mathrel{\widehat{=}} T \geq a_3 c^2 - 4.5c + 9 \wedge c \geq 0 \wedge c \leq 1$
- By the second step, we get

$$10a_3 - 9 \leq 0 \wedge 2a_3 - 1 \geq 0 \wedge a_1 + 2 = 0 \wedge a_0 + 2a_1 + 9 = 0 \wedge a_2 - a_0 = 0\,.$$

- By choosing $a_0 = -5, a_1 = -2, a_2 = -5, a_3 = \frac{1}{2}$, obtain the following SGI
  - $I_{q_{\mathrm{ht}}} \mathrel{\widehat{=}} T \geq 2c + 5 \wedge c \geq 0$;
  - $I_{q_{\mathrm{cl}}} \mathrel{\widehat{=}} T \geq 5$;
  - $I_{q_{\mathrm{ck}}} \mathrel{\widehat{=}} 2T \geq c^2 - 9c + 18 \wedge c \geq 0 \wedge c \leq 1$.
- **The safety property is successfully verified by the SGI.**

Running Example (Cont'd)

- Firstly, predefine the following set of templates:
  - $I_{q_{\mathrm{ht}}} \mathrel{\widehat{=}} T + a_1 c + a_0 \geq 0 \wedge c \geq 0$;
  - $I_{q_{\mathrm{cl}}} \mathrel{\widehat{=}} T + a_2 \geq 0$;
  - $I_{q_{\mathrm{ck}}} \mathrel{\widehat{=}} T \geq a_3 c^2 - 4.5c + 9 \wedge c \geq 0 \wedge c \leq 1$
- By the second step, we get

  $10a_3 - 9 \leq 0 \wedge 2a_3 - 1 \geq 0 \wedge a_1 + 2 = 0 \wedge a_0 + 2a_1 + 9 = 0 \wedge a_2 - a_0 = 0\,.$

- By choosing $a_0 = -5, a_1 = -2, a_2 = -5, a_3 = \frac{1}{2}$, obtain the following SGI
  - $I_{q_{\mathrm{ht}}} \mathrel{\widehat{=}} T \geq 2c + 5 \wedge c \geq 0$;
  - $I_{q_{\mathrm{cl}}} \mathrel{\widehat{=}} T \geq 5$;
  - $I_{q_{\mathrm{ck}}} \mathrel{\widehat{=}} 2T \geq c^2 - 9c + 18 \wedge c \geq 0 \wedge c \leq 1$.
- The safety property is successfully verified by the SGI.

Running Example (Cont'd)

- Firstly, predefine the following set of templates:
  - $I_{q_{\rm ht}} \mathrel{\widehat{=}} T + a_1 c + a_0 \geq 0 \wedge c \geq 0$;
  - $I_{q_{\rm cl}} \mathrel{\widehat{=}} T + a_2 \geq 0$;
  - $I_{q_{\rm ck}} \mathrel{\widehat{=}} T \geq a_3 c^2 - 4.5c + 9 \wedge c \geq 0 \wedge c \leq 1$
- By the second step, we get

  $$10a_3 - 9 \leq 0 \wedge 2a_3 - 1 \geq 0 \wedge a_1 + 2 = 0 \wedge a_0 + 2a_1 + 9 = 0 \wedge a_2 - a_0 = 0\,.$$

- By choosing $a_0 = -5, a_1 = -2, a_2 = -5, a_3 = \frac{1}{2}$, obtain the following SGI
  - $I_{q_{\rm ht}} \mathrel{\widehat{=}} T \geq 2c + 5 \wedge c \geq 0$;
  - $I_{q_{\rm cl}} \mathrel{\widehat{=}} T \geq 5$;
  - $I_{q_{\rm ck}} \mathrel{\widehat{=}} 2T \geq c^2 - 9c + 18 \wedge c \geq 0 \wedge c \leq 1$.
- The safety property is successfully verified by the SGI.

Running Example (Cont'd)

- Firstly, predefine the following set of templates:
  - $I_{q_{\mathrm{ht}}} \mathrel{\widehat{=}} T + a_1 c + a_0 \geq 0 \wedge c \geq 0$;
  - $I_{q_{\mathrm{cl}}} \mathrel{\widehat{=}} T + a_2 \geq 0$;
  - $I_{q_{\mathrm{ck}}} \mathrel{\widehat{=}} T \geq a_3 c^2 - 4.5c + 9 \wedge c \geq 0 \wedge c \leq 1$
- By the second step, we get

$$10a_3 - 9 \leq 0 \wedge 2a_3 - 1 \geq 0 \wedge a_1 + 2 = 0 \wedge a_0 + 2a_1 + 9 = 0 \wedge a_2 - a_0 = 0\,.$$

- By choosing $a_0 = -5, a_1 = -2, a_2 = -5, a_3 = \frac{1}{2}$, obtain the following SGI
  - $I_{q_{\mathrm{ht}}} \mathrel{\widehat{=}} T \geq 2c + 5 \wedge c \geq 0$;
  - $I_{q_{\mathrm{cl}}} \mathrel{\widehat{=}} T \geq 5$;
  - $I_{q_{\mathrm{ck}}} \mathrel{\widehat{=}} 2T \geq c^2 - 9c + 18 \wedge c \geq 0 \wedge c \leq 1$.
- **The safety property is successfully verified by the SGI.**

Outline

Outline

Problem Decription

- A safety requirement $\mathcal{S}$ assigns to each mode $q \in Q$ a safe region $S_q \subseteq \mathbb{R}^n$, i.e. $\mathcal{S} = \bigcup_{q \in Q}(\{q\} \times S_q)$.

Switching controller synthesis for safety [Asarin et al. 00]

Given a hybrid automaton $\mathcal{H}$ and a safety property S, find a hybrid automaton $\mathcal{H}' = (Q, X, f, D', E, G')$ such that

(r1) Refinement: for any $q \in Q$, $D'_q \subseteq D_q$, and for any $e \in E$, $G'_e \subseteq G_e$;

(r2) Safety: for any trajectory ω that $\mathcal{H}'$ accepts, if (q, x) is on ω, then $\mathrm{x} \in S_q$;

(r3) Non-blocking: $\mathcal{H}'$ is non-blocking.

Problem Decription

- A safety requirement $\mathcal{S}$ assigns to each mode $q \in Q$ a safe region $S_q \subseteq \mathbb{R}^n$, i.e. $\mathcal{S} = \bigcup_{q \in Q}(\{q\} \times S_q)$.

Switching controller synthesis for safety [Asarin et al. 00]

Given a hybrid automaton $\mathcal{H}$ and a safety property S, find a hybrid automaton $\mathcal{H}' = (Q, X, f, D', E, G')$ such that

(r1) Refinement: for any $q \in Q$, $D'_q \subseteq D_q$, and for any $e \in E$, $G'_e \subseteq G_e$;

(r2) Safety: for any trajectory ω that $\mathcal{H}'$ accepts, if (q, x) is on ω, then $x \in S_q$;

(r3) Non-blocking: $\mathcal{H}'$ is non-blocking.

Problem Decription

- A safety requirement $\mathcal{S}$ assigns to each mode $q \in Q$ a safe region $S_q \subseteq \mathbb{R}^n$, i.e. $\mathcal{S} = \bigcup_{q \in Q}(\{q\} \times S_q)$.

Switching controller synthesis for safety [Asarin et al. 00]

Given a hybrid automaton $\mathcal{H}$ and a safety property S, find a hybrid automaton $\mathcal{H}' = (Q, X, f, D', E, G')$ such that

(r1) Refinement: for any $q \in Q$, $D'_q \subseteq D_q$, and for any $e \in E$, $G'_e \subseteq G_e$;

(r2) Safety: for any trajectory ω that $\mathcal{H}'$ accepts, if $(q, \mathbf{x})$ is on ω, then $\mathbf{x} \in S_q$;

(r3) Non-blocking: $\mathcal{H}'$ is non-blocking.

Problem Decription

- A safety requirement $\mathcal{S}$ assigns to each mode $q \in Q$ a safe region $S_q \subseteq \mathbb{R}^n$, i.e. $\mathcal{S} = \bigcup_{q \in Q}(\{q\} \times S_q)$.

Switching controller synthesis for safety [Asarin et al. 00]

Given a hybrid automaton $\mathcal{H}$ and a safety property S, find a hybrid automaton $\mathcal{H}' = (Q, X, f, D', E, G')$ such that

(r1) Refinement: for any $q \in Q$, $D'_q \subseteq D_q$, and for any $e \in E$, $G'_e \subseteq G_e$;

(r2) Safety: for any trajectory ω that $\mathcal{H}'$ accepts, if $(q, \mathbf{x})$ is on ω, then $\mathbf{x} \in S_q$;

(r3) Non-blocking: $\mathcal{H}'$ is non-blocking.

Problem Decription

- A safety requirement $\mathcal{S}$ assigns to each mode $q \in Q$ a safe region $S_q \subseteq \mathbb{R}^n$, i.e. $\mathcal{S} = \bigcup_{q \in Q}(\{q\} \times S_q)$.

Switching controller synthesis for safety [Asarin et al. 00]

Given a hybrid automaton $\mathcal{H}$ and a safety property S, find a hybrid automaton $\mathcal{H}' = (Q, X, f, D', E, G')$ such that

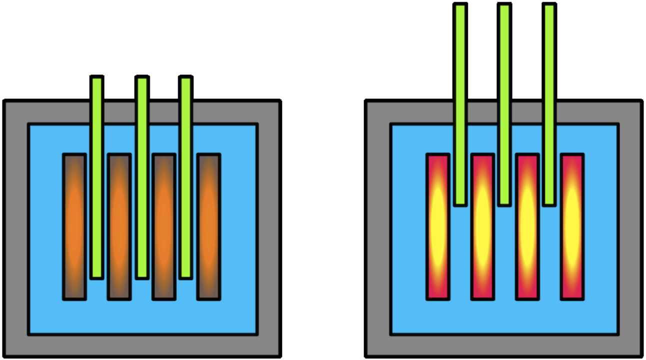
(r1) Refinement: for any $q \in Q$, $D'_q \subseteq D_q$, and for any $e \in E$, $G'_e \subseteq G_e$;

(r2) Safety: for any trajectory ω that $\mathcal{H}'$ accepts, if $(q, \mathbf{x})$ is on ω, then $\mathbf{x} \in S_q$;

(r3) Non-blocking: $\mathcal{H}'$ is non-blocking.

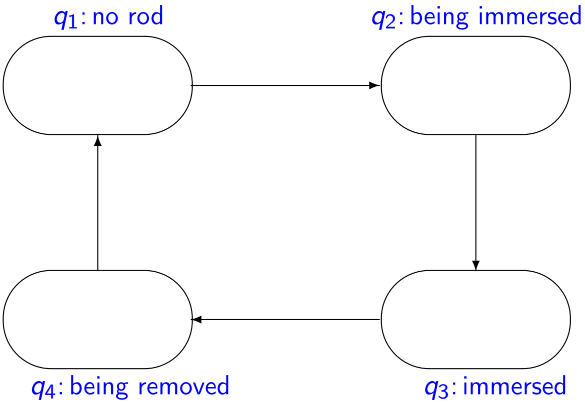
A Nuclear Reactor Example

The nuclear reactor system consists of a reactor core and a cooling rod which is immersed into and removed out of the core periodically to keep the temperature of the core in a certain range.



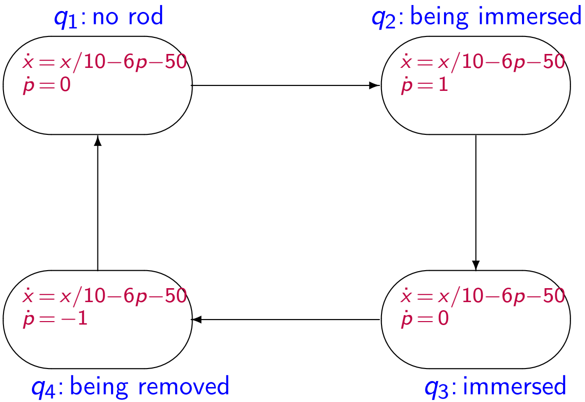
A Nuclear Reactor Example (Cont'd)

- x: temperature;
- p: proportion immersed



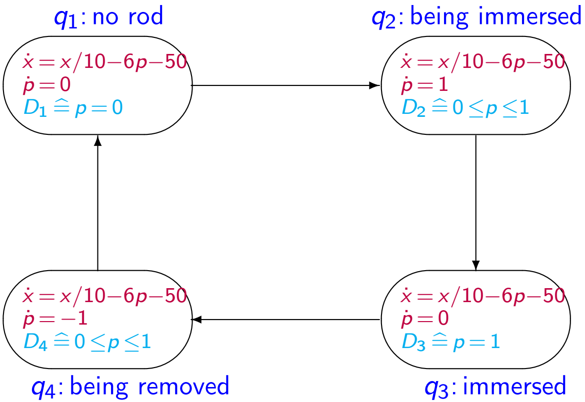
A Nuclear Reactor Example (Cont'd)

- x: temperature;
- p: proportion immersed



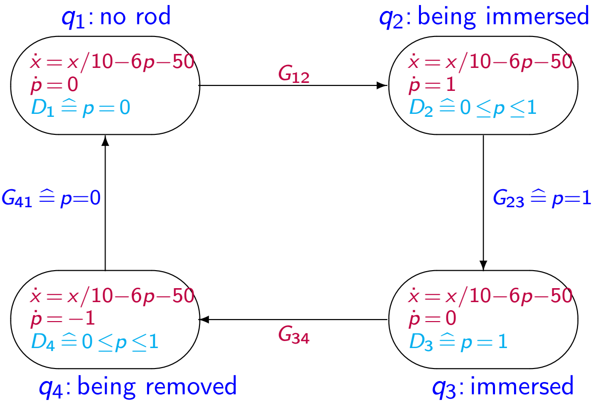
A Nuclear Reactor Example (Cont'd)

- x: temperature;
- p: proportion immersed



A Nuclear Reactor Example (Cont'd)

- x: temperature;
- p: proportion immersed

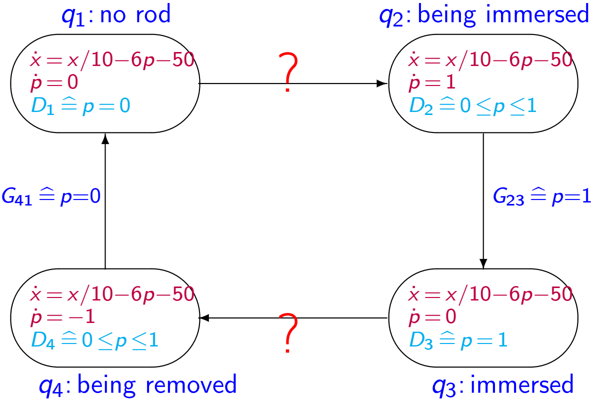


Switching Controller Synthesis for the Reactor

$S \mathrel{\widehat{=}} 510 \leq x \leq 550$ for all modes

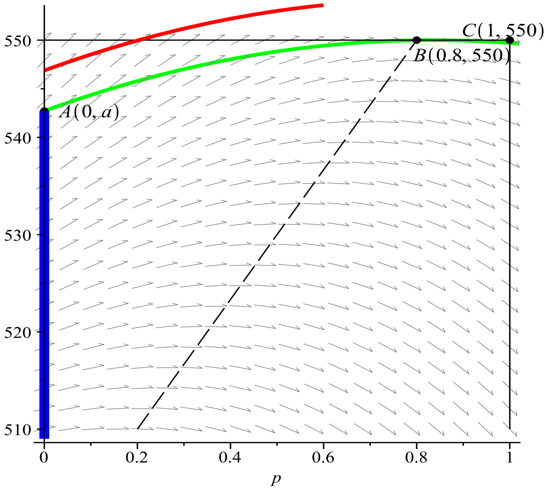
Switching Controller Synthesis for the Reactor

$S \mathrel{\widehat{=}} 510 \leq x \leq 550$ for all modes



Bad Switching Violates Safety Property

Transition from mode q_1 to q_2



Solution to the Controller Synthesis Problem

Abstract Solution

Let $\mathcal{H}$ be a hybrid system and $\mathcal{S}$ be a safety property. If we can find a family of $D'_q \subseteq \mathbb{R}^n$ such that

(c1) for all $q \in Q$, $D'_q \subseteq D_q \cap S_q$;

(c2) for all $q \in Q$, D'_q is a continuous invariant of $(H_q, \mathbf{f}_q)$ with

$$H_q \widehat{=} \Big(\bigcup_{e=(q,q')\in E} G'_e \Big)^c ,$$

where $G'_e \widehat{=} G_e \cap D'_{q'}$ for $e = (q, q')$, then the family of G'_e form a safe switching controller.

Solution to the Controller Synthesis Problem

Abstract Solution

Let $\mathcal{H}$ be a hybrid system and $\mathcal{S}$ be a safety property. If we can find a family of $D'_q \subseteq \mathbb{R}^n$ such that

(c1) for all $q \in Q$, $D'_q \subseteq D_q \cap S_q$;

(c2) for all $q \in Q$, D'_q is a continuous invariant of $(H_q, \mathbf{f}_q)$ with

$$H_q \mathrel{\widehat{=}} \Big(\bigcup_{e=(q,q')\in E} G'_e \Big)^c ,$$

where $G'_e \mathrel{\widehat{=}} G_e \cap D'_{q'}$ for $e = (q, q')$, then the family of G'_e form a safe switching controller.

Solution to the Controller Synthesis Problem

Abstract Solution

Let $\mathcal{H}$ be a hybrid system and $\mathcal{S}$ be a safety property. If we can find a family of $D'_q \subseteq \mathbb{R}^n$ such that

(c1) for all $q \in Q$, $D'_q \subseteq D_q \cap S_q$;

(c2) for all $q \in Q$, D'_q is a continuous invariant of $(H_q, \mathbf{f}_q)$ with

$$H_q \mathrel{\widehat{=}} \Big(\bigcup_{e=(q,q')\in E} G'_e \Big)^c ,$$

where $G'_e \mathrel{\widehat{=}} G_e \cap D'_{q'}$ for $e = (q, q')$, then the family of G'_e form a safe switching controller.

Template-Based Synthesis Framework

(s1) Template assignment: assign to each $q \in Q$ a template D'_q as the continuous invariant to be generated at mode q ;

(s2) Guard refinement: refine the transition guard G_e for each $e = (q, q') \in E$ by setting $G'_e \mathrel{\widehat{=}} G_e \cap D'_{q'}$;

(s3) Deriving synthesis conditions: encode (c1) and (c2) in the abstract solution into constraints on parameters appearing in the templates;

(s4) Constraint solving: solve the constraints derived from (s3) using quantifier elimination (QE);

(s5) Parameters instantiation: find an appropriate instantiation of D'_q and G'_e from the possible parameter values obtained at (s4)

Template-Based Synthesis Framework

(s1) Template assignment: assign to each $q \in Q$ a template D'_q as the continuous invariant to be generated at mode q;

(s2) Guard refinement: refine the transition guard G_e for each $e = (q, q') \in E$ by setting $G'_e \mathrel{\widehat{=}} G_e \cap D'_{q'}$;

(s3) Deriving synthesis conditions: encode (c1) and (c2) in the abstract solution into constraints on parameters appearing in the templates;

(s4) Constraint solving: solve the constraints derived from (s3) using quantifier elimination (QE);

(s5) Parameters instantiation: find an appropriate instantiation of D'_q and G'_e from the possible parameter values obtained at (s4)

Template-Based Synthesis Framework

(s1) Template assignment: assign to each $q \in Q$ a template D'_q as the continuous invariant to be generated at mode q ;

(s2) Guard refinement: refine the transition guard G_e for each $e = (q, q') \in E$ by setting $G'_e \mathrel{\widehat{=}} G_e \cap D'_{q'}$;

(s3) Deriving synthesis conditions: encode (c1) and (c2) in the abstract solution into constraints on parameters appearing in the templates;

(s4) Constraint solving: solve the constraints derived from (s3) using quantifier elimination (QE);

(s5) Parameters instantiation: find an appropriate instantiation of D'_q and G'_e from the possible parameter values obtained at (s4)

Template-Based Synthesis Framework

(s1) Template assignment: assign to each $q \in Q$ a template D'_q as the continuous invariant to be generated at mode q ;

(s2) Guard refinement: refine the transition guard G_e for each $e = (q, q') \in E$ by setting $G'_e \widehat{=} G_e \cap D'_{q'}$;

(s3) Deriving synthesis conditions: encode (c1) and (c2) in the abstract solution into constraints on parameters appearing in the templates;

(s4) Constraint solving: solve the constraints derived from (s3) using quantifier elimination (QE);

(s5) Parameters instantiation: find an appropriate instantiation of D'_q and G'_e from the possible parameter values obtained at (s4)

Template-Based Synthesis Framework

(s1) Template assignment: assign to each $q \in Q$ a template D'_q as the continuous invariant to be generated at mode q;

(s2) Guard refinement: refine the transition guard G_e for each $e = (q, q') \in E$ by setting $G'_e \widehat{=} G_e \cap D'_{q'}$;

(s3) Deriving synthesis conditions: encode (c1) and (c2) in the abstract solution into constraints on parameters appearing in the templates;

(s4) Constraint solving: solve the constraints derived from (s3) using quantifier elimination (QE);

(s5) Parameters instantiation: find an appropriate instantiation of D'_q and G'_e from the possible parameter values obtained at (s4)

Heuristics for Predefining Templates by Qualitative Analysis

Using qualitative analysis to identify critical points for predefining templates

- Infer the evolution behavior (increasing or decreasing) of continuous variables in each mode from the ODEs
- Identify modes (called critical) at which the evolution behavior of a continuous variable changes, and thus the maximal (or minimal) value of this continuous variable can be achieved
- Equate the maximal (or minimal) value to the corresponding safety upper (or lower) bound to obtain a critical point
- Backward propagate the critical point, by backtracking along the continuous trajectory through the critical point

Heuristics for Predefining Templates by Qualitative Analysis

Using qualitative analysis to identify critical points for predefining templates

- Infer the evolution behavior (increasing or decreasing) of continuous variables in each mode from the ODEs
- Identify modes (called critical) at which the evolution behavior of a continuous variable changes, and thus the maximal (or minimal) value of this continuous variable can be achieved
- Equate the maximal (or minimal) value to the corresponding safety upper (or lower) bound to obtain a critical point
- Backward propagate the critical point, by backtracking along the continuous trajectory through the critical point

Heuristics for Predefining Templates by Qualitative Analysis

Using qualitative analysis to identify critical points for predefining templates

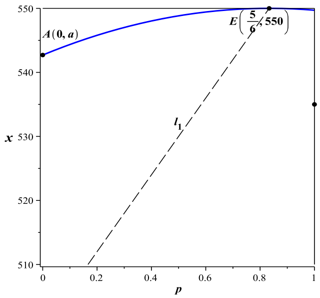
- Infer the evolution behavior (increasing or decreasing) of continuous variables in each mode from the ODEs
- Identify modes (called critical) at which the evolution behavior of a continuous variable changes, and thus the maximal (or minimal) value of this continuous variable can be achieved
- Equate the maximal (or minimal) value to the corresponding safety upper (or lower) bound to obtain a critical point
- Backward propagate the critical point, by backtracking along the continuous trajectory through the critical point

Heuristics for Predefining Templates by Qualitative Analysis

Using qualitative analysis to identify critical points for predefining templates

- Infer the evolution behavior (increasing or decreasing) of continuous variables in each mode from the ODEs
- Identify modes (called critical) at which the evolution behavior of a continuous variable changes, and thus the maximal (or minimal) value of this continuous variable can be achieved
- Equate the maximal (or minimal) value to the corresponding safety upper (or lower) bound to obtain a critical point
- Backward propagate the critical point, by backtracking along the continuous trajectory through the critical point

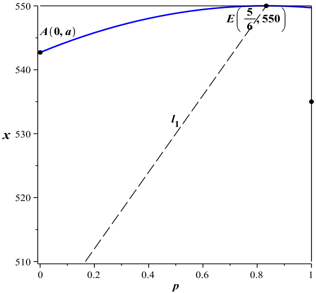
Revisiting the Running Example



For the running example,

- At D_{q_2}, temperature x achieves maximal value when crossing $l_1 \widehat{=} x/10 - 6p - 50 = 0$.
- $E(5/6, 550)$ at q_2 is obtained by taking the intersection of l_1 and safety upper bound $x = 550$
- E is backward propagated to $A(0, a)$, with a a parameter
- Compute a parabola $x - 550 - \frac{36}{25}(a - 550)(p - \frac{5}{6})^2 = 0$ through A and E as part of the template D'_{q_2}

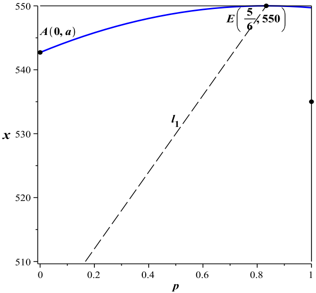
Revisiting the Running Example



For the running example,

- At D_{q_2}, temperature x achieves maximal value when crossing $l_1 \widehat{=} x/10 - 6p - 50 = 0$.
- $E(5/6, 550)$ at q_2 is obtained by taking the intersection of l_1 and safety upper bound $x = 550$
- E is backward propagated to $A(0, a)$, with a a parameter
- Compute a parabola $x - 550 - \frac{36}{25}(a - 550)(p - \frac{5}{6})^2 = 0$ through A and E as part of the template D'_{q_2}

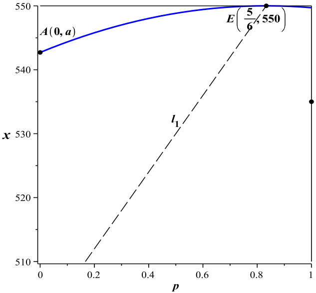
Revisiting the Running Example



For the running example,

- At D_{q_2}, temperature x achieves maximal value when crossing $l_1 \widehat{=} x/10 - 6p - 50 = 0$.
- $E(5/6, 550)$ at q_2 is obtained by taking the intersection of l_1 and safety upper bound $x = 550$
- E is backward propagated to $A(0, a)$, with a a parameter
- Compute a parabola $x - 550 - \frac{36}{25}(a - 550)(p - \frac{5}{6})^2 = 0$ through A and E as part of the template D'_{q_2}

Revisiting the Running Example



For the running example,

- At D_{q_2}, temperature x achieves maximal value when crossing $l_1 \widehat{=} x/10 - 6p - 50 = 0$.
- $E(5/6, 550)$ at q_2 is obtained by taking the intersection of l_1 and safety upper bound $x = 550$
- E is backward propagated to $A(0, a)$, with a a parameter
- Compute a parabola $x - 550 - \frac{36}{25}(a - 550)(p - \frac{5}{6})^2 = 0$ through A and E as part of the template D'_{q_2}

Revisiting the Running Example (Cont'd)

The set of parameters: a, b, c, d

- $D'_1 \hat{=}\ p = 0 \wedge 510 \leq x \leq a$
- $D'_2 \hat{=}\ 0 \leq p \leq 1 \wedge x - b \geq p(d-b) \wedge$
  $x - 550 - \frac{36}{25}(a-550)(p-\frac{5}{6})^2 \leq 0$
- $D'_3 \hat{=}\ p = 1 \wedge d \leq x \leq 550$
- $D'_4 \hat{=}\ 0 \leq p \leq 1 \wedge x - a \leq p(c-a) \wedge$
  $x - 510 - \frac{36}{25}(d-510)(p-\frac{1}{6})^2 \geq 0$

- $G'_{12} \hat{=}\ p = 0 \wedge b \leq x \leq a$
- $G'_{23} \hat{=}\ p = 1 \wedge d \leq x \leq 550$
- $G'_{34} \hat{=}\ p = 1 \wedge d \leq x \leq c$
- $G'_{41} \hat{=}\ p = 0 \wedge 510 \leq x \leq a$

Revisiting the Running Example (Cont'd)

The set of parameters: a, b, c, d

- $D_1' \mathrel{\widehat{=}} p = 0 \wedge 510 \leq x \leq a$
- $D_2' \mathrel{\widehat{=}} 0 \leq p \leq 1 \wedge x - b \geq p(d - b) \wedge$
  $x - 550 - \frac{36}{25}(a - 550)(p - \frac{5}{6})^2 \leq 0$
- $D_3' \mathrel{\widehat{=}} p = 1 \wedge d \leq x \leq 550$
- $D_4' \mathrel{\widehat{=}} 0 \leq p \leq 1 \wedge x - a \leq p(c - a) \wedge$
  $x - 510 - \frac{36}{25}(d - 510)(p - \frac{1}{6})^2 \geq 0$

- $G_{12}' \mathrel{\widehat{=}} p = 0 \wedge b \leq x \leq a$
- $G_{23}' \mathrel{\widehat{=}} p = 1 \wedge d \leq x \leq 550$
- $G_{34}' \mathrel{\widehat{=}} p = 1 \wedge d \leq x \leq c$
- $G_{41}' \mathrel{\widehat{=}} p = 0 \wedge 510 \leq x \leq a$

Revisiting the Running Example (Cont'd)

- $a = \frac{6575}{12} \wedge b = \frac{4135}{8} \wedge c = \frac{4345}{8} \wedge d = \frac{6145}{12}$.
- From this result we get that the cooling rod should be immersed before temperature rises to $\frac{6575}{12} = 547.92$, and removed before temperature drops to $\frac{6145}{12} = 512.08$.
- By solving differential equations explicitly, the corresponding exact bounds are 547.97 and 512.03

Revisiting the Running Example (Cont'd)

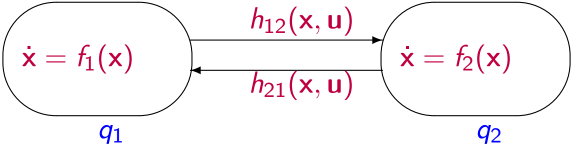
- $a = \frac{6575}{12} \wedge b = \frac{4135}{8} \wedge c = \frac{4345}{8} \wedge d = \frac{6145}{12}$.
- From this result we get that the cooling rod should be immersed before temperature rises to $\frac{6575}{12} = 547.92$, and removed before temperature drops to $\frac{6145}{12} = 512.08$.
- By solving differential equations explicitly, the corresponding exact bounds are 547.97 and 512.03

Revisiting the Running Example (Cont'd)

- $a = \frac{6575}{12} \wedge b = \frac{4135}{8} \wedge c = \frac{4345}{8} \wedge d = \frac{6145}{12}$.
- From this result we get that the cooling rod should be immersed before temperature rises to $\frac{6575}{12} = 547.92$, and removed before temperature drops to $\frac{6145}{12} = 512.08$.
- By solving differential equations explicitly, the corresponding exact bounds are 547.97 and 512.03

Outline

Problem Description



- Given a hybrid system $\mathcal{H}$ in which transition conditions h_{ij} are not determined but parameterized by $\mathbf{u}$, a vector of control parameters
- Our task is to determine $\mathbf{u}$ such that $\mathcal{H}$ can make discrete jumps at desired points, thus guaranteeing that
  - a safety property $\mathcal{S}$ is satisfied, i.e. $\mathbf{x} \in \mathcal{S}$ at any time
  - an optimization goal, e.g. $\min_{\mathbf{u}} g(\mathbf{u})$, is achieved

Our Approach – Step 1

Derive constraint $D(\mathbf{u})$ on $\mathbf{u}$ from the safety requirements $\mathcal{S}$

- Compute
  - the exact reachable set $\mathrm{Reach}_{\mathcal{H}}(\mathbf{x}, \mathbf{u})$ of $\mathcal{H}$, or
  - an inductive invariant $\mathrm{Inv}_{\mathcal{H}}(\mathbf{x}, \mathbf{u})$

  as polynomial formulas
- Suppose $\mathcal{S}$ is also modeled by polynomial formulas, then $D(\mathbf{u})$ can be obtained by applying QE to

$$\forall \mathbf{x}.\ \Big(\mathrm{Reach}_{\mathcal{H}}(\mathbf{x}, \mathbf{u}) \longrightarrow \mathcal{S}\Big)$$

or

$$\forall \mathbf{x}.\ \Big(\mathrm{Inv}_{\mathcal{H}}(\mathbf{x}, \mathbf{u}) \longrightarrow \mathcal{S}\Big)$$

Our Approach – Step 1

Derive constraint $D(\mathbf{u})$ on $\mathbf{u}$ from the safety requirements $\mathcal{S}$

- Compute
  - the exact reachable set $\mathrm{Reach}_{\mathcal{H}}(\mathbf{x}, \mathbf{u})$ of $\mathcal{H}$, or
  - an inductive invariant $\mathrm{Inv}_{\mathcal{H}}(\mathbf{x}, \mathbf{u})$

  as polynomial formulas
- Suppose $\mathcal{S}$ is also modeled by polynomial formulas, then $D(\mathbf{u})$ can be obtained by applying QE to

$$\forall \mathbf{x}.\ \left(\mathrm{Reach}_{\mathcal{H}}(\mathbf{x}, \mathbf{u}) \longrightarrow \mathcal{S}\right)$$

or

$$\forall \mathbf{x}.\ \left(\mathrm{Inv}_{\mathcal{H}}(\mathbf{x}, \mathbf{u}) \longrightarrow \mathcal{S}\right)$$

Our Approach – Step 1

Derive constraint $D(\mathbf{u})$ on $\mathbf{u}$ from the safety requirements $\mathcal{S}$

- Compute
  - the exact reachable set $\mathrm{Reach}_{\mathcal{H}}(\mathbf{x}, \mathbf{u})$ of $\mathcal{H}$, or
  - an inductive invariant $\mathrm{Inv}_{\mathcal{H}}(\mathbf{x}, \mathbf{u})$

  as polynomial formulas
- Suppose $\mathcal{S}$ is also modeled by polynomial formulas, then $D(\mathbf{u})$ can be obtained by applying QE to

$$\forall \mathbf{x}. \left(\mathrm{Reach}_{\mathcal{H}}(\mathbf{x}, \mathbf{u}) \longrightarrow \mathcal{S}\right)$$

or

$$\forall \mathbf{x}. \left(\mathrm{Inv}_{\mathcal{H}}(\mathbf{x}, \mathbf{u}) \longrightarrow \mathcal{S}\right)$$

Our Approach – Step 1

Derive constraint $D(\mathbf{u})$ on $\mathbf{u}$ from the safety requirements $\mathcal{S}$

- Compute
  - the exact reachable set $\mathrm{Reach}_{\mathcal{H}}(\mathbf{x}, \mathbf{u})$ of $\mathcal{H}$, or
  - an inductive invariant $\mathrm{Inv}_{\mathcal{H}}(\mathbf{x}, \mathbf{u})$

  as polynomial formulas
- Suppose $\mathcal{S}$ is also modeled by polynomial formulas, then $D(\mathbf{u})$ can be obtained by applying QE to

$$\forall \mathbf{x}.\ \Big(\mathrm{Reach}_{\mathcal{H}}(\mathbf{x}, \mathbf{u}) \longrightarrow \mathcal{S}\Big)$$

or

$$\forall \mathbf{x}.\ \Big(\mathrm{Inv}_{\mathcal{H}}(\mathbf{x}, \mathbf{u}) \longrightarrow \mathcal{S}\Big)$$

Our Approach – Step 2

Encode the optimization problem (suppose the objective function g is a polynomial) over constraint $D(\mathbf{u})$ into a quantified first-order polynomial formula $\mathbf{Qu}.\varphi(\mathbf{u}, z)$ by introducing a fresh variable z

- Minimize u^2 on $[-1, 1]$
- Introduce a fresh variable z:
  $u \geq -1 \wedge u \leq 1 \wedge\ u^2 \leq z$
- Projection to the z-axis:
  $\exists u.(u \geq -1 \wedge u \leq 1 \wedge u^2 \leq z)$
- After QE: $z \geq 0$, which means

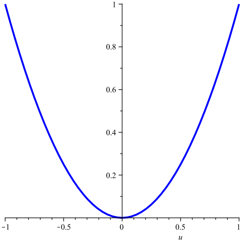
$$\min_{u \in [-1,1]} u^2 = 0$$

Our Approach – Step 2

Encode the optimization problem (suppose the objective function g is a polynomial) over constraint $D(\mathbf{u})$ into a quantified first-order polynomial formula $\mathbf{Qu}.\varphi(\mathbf{u}, z)$ by introducing a fresh variable z

- Minimize u^2 on $[-1, 1]$
- Introduce a fresh variable z:
  $u \geq -1 \wedge u \leq 1 \wedge \; u^2 \leq z$
- Projection to the z-axis:
  $\exists u.(u \geq -1 \wedge u \leq 1 \wedge u^2 \leq z)$
- After QE: $z \geq 0$, which means

$$\min_{u \in [-1,1]} u^2 = 0$$

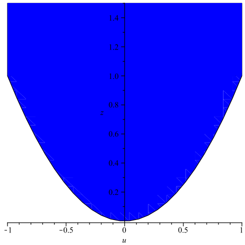


Our Approach – Step 2

Encode the optimization problem (suppose the objective function g is a polynomial) over constraint $D(\mathbf{u})$ into a quantified first-order polynomial formula $\mathbf{Qu}.\varphi(\mathbf{u}, z)$ by introducing a fresh variable z

- Minimize u^2 on $[-1, 1]$
- Introduce a fresh variable z:
  $u \geq -1 \wedge u \leq 1 \wedge\ u^2 \leq z$
- Projection to the z-axis:
  $\exists u.(u \geq -1 \wedge u \leq 1 \wedge u^2 \leq z)$
- After QE: $z \geq 0$, which means

$$\min_{u \in [-1,1]} u^2 = 0$$

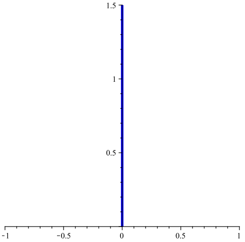


Our Approach – Step 2

Encode the optimization problem (suppose the objective function g is a polynomial) over constraint $D(\mathbf{u})$ into a quantified first-order polynomial formula $\mathbf{Qu}.\varphi(\mathbf{u}, z)$ by introducing a fresh variable z

- Minimize u^2 on $[-1, 1]$
- Introduce a fresh variable z:
  $u \geq -1 \wedge u \leq 1 \wedge\ u^2 \leq z$
- Projection to the z-axis:
  $\exists u.(u \geq -1 \wedge u \leq 1 \wedge u^2 \leq z)$
- After QE: $z \geq 0$, which means

$$\min_{u \in [-1,1]} u^2 = 0$$

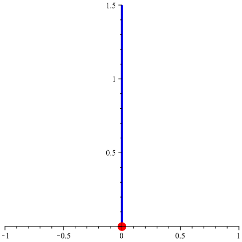


Our Approach – Step 2

Encode the optimization problem (suppose the objective function g is a polynomial) over constraint $D(\mathbf{u})$ into a quantified first-order polynomial formula $\mathbf{Qu}.\varphi(\mathbf{u}, z)$ by introducing a fresh variable z

- Minimize u^2 on $[-1, 1]$
- Introduce a fresh variable z:
  $u \geq -1 \wedge u \leq 1 \wedge \; u^2 \leq z$
- Projection to the z-axis:
  $\exists u.(u \geq -1 \wedge u \leq 1 \wedge u^2 \leq z)$
- After QE: $z \geq 0$, which means

$$\min_{u \in [-1,1]} u^2 = 0$$



Encoding Optimization Criteria

Lemma

Suppose $g_1(\mathbf{u}_1)$, $g_2(\mathbf{u}_1, \mathbf{u}_2)$, $g_3(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ *are polynomials, and* $D_1(\mathbf{u}_1)$, $D_2(\mathbf{u}_1, \mathbf{u}_2)$, $D_3(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ *are nonempty compact semi-algebraic sets. Then there exist* c_1, c_2, $c_3 \in \mathbb{R}$ *s.t.*

$$\exists \mathbf{u}_1.(D_1 \wedge g_1 \leq z) \Leftrightarrow z \geq c_1 \tag{4}$$

$$\forall \mathbf{u}_2.\big(\exists \mathbf{u}_1.D_2 \Rightarrow \exists \mathbf{u}_1.(D_2 \wedge g_2 \leq z)\big) \Leftrightarrow z \geq c_2 \tag{5}$$

$$\exists \mathbf{u}_3.\Big((\exists \mathbf{u}_1 \mathbf{u}_2.D_3) \wedge \forall \mathbf{u}_2.\big(\exists \mathbf{u}_1.D_3 \Rightarrow \exists \mathbf{u}_1.(D_3 \wedge g_3 \leq z)\big)\Big) \Leftrightarrow z \rhd c_3 \tag{6}$$

where $\rhd \in \{>, \geq\}$, *and* c_1, c_2, c_3 *satisfy*

$$c_1 = \min_{\mathbf{u}_1} g_1(\mathbf{u}_1) \quad \text{over } D_1(\mathbf{u}_1)\,, \tag{7}$$

$$c_2 = \sup_{\mathbf{u}_2} \min_{\mathbf{u}_1} g_2(\mathbf{u}_1, \mathbf{u}_2) \quad \text{over } D_2(\mathbf{u}_1, \mathbf{u}_2)\,, \tag{8}$$

$$c_3 = \inf_{\mathbf{u}_3} \sup_{\mathbf{u}_2} \min_{\mathbf{u}_1} g_3(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3) \quad \text{over } D_3(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)\,. \tag{9}$$

Our Approach – Step 3

Eliminate quantifiers in $\mathbf{Qu}.\varphi(\mathbf{u}, z)$ and from the result we can retrieve the optimal value and the corresponding optimal controller $\mathbf{u}$

- Combine exact QE with numeric computation: (discretization of existentially quantified variables)

$$\exists \mathbf{x} \in A.\, \varphi(\mathbf{x}) \approx \bigvee_{\mathbf{y} \in F_A} \varphi(\mathbf{y}) \quad ,$$

where F_A is a finite subset of A

Our Approach – Step 3

Eliminate quantifiers in $\mathbf{Q}\mathbf{u}.\varphi(\mathbf{u}, z)$ and from the result we can retrieve the optimal value and the corresponding optimal controller $\mathbf{u}$

- Combine exact QE with numeric computation: (discretization of existentially quantified variables)

$$\exists \mathbf{x} \in A.\, \varphi(\mathbf{x}) \approx \bigvee_{\mathbf{y} \in F_A} \varphi(\mathbf{y}) \quad ,$$

where F_A is a finite subset of A

Outline

A Reported Case Study

Cassez, F., Jessen, J.J., Larsen, K.G., Raskin, J.F., Reynier, P.A.:
Automatic Synthesis of Robust and Optimal Controllers — An Industrial Case Study. HSCC'09

- Provided by the HYDAC ELECTRONIC GMBH company within the European project *Quasimodo*
- An oil pump control problem
  - safety
  - robustness
  - optimality

A Reported Case Study

Cassez, F., Jessen, J.J., Larsen, K.G., Raskin, J.F., Reynier, P.A.:
Automatic Synthesis of Robust and Optimal Controllers — An Industrial Case Study. HSCC'09

- Provided by the HYDAC ELECTRONIC GMBH company within the European project *Quasimodo*
- An oil pump control problem
  - safety
  - robustness
  - optimality

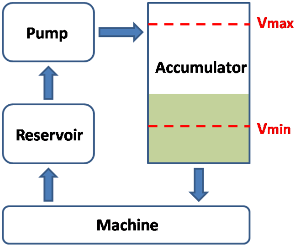
A Reported Case Study

Cassez, F., Jessen, J.J., Larsen, K.G., Raskin, J.F., Reynier, P.A.:
Automatic Synthesis of Robust and Optimal Controllers — An Industrial Case Study. HSCC'09

- Provided by the HYDAC ELECTRONIC GMBH company within the European project *Quasimodo*
- An oil pump control problem
  - safety
  - robustness
  - optimality

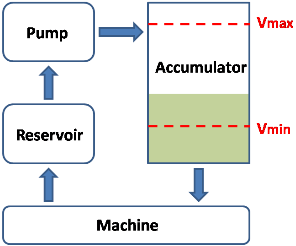
The System

- The system is composed of a machine, an accumulator, a reservoir and a pump
- The machine consumes oil out of the accumulator; the pump adds oil from the reservoir into the accumulator



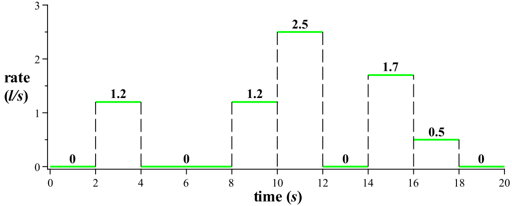
The System

- The system is composed of a machine, an accumulator, a reservoir and a pump
- The machine consumes oil out of the accumulator; the pump adds oil from the reservoir into the accumulator



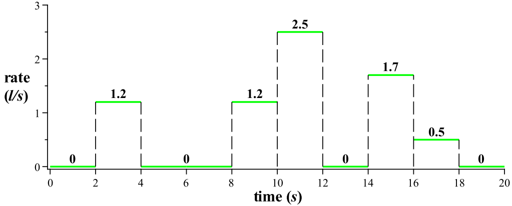
The Consumption Rate

- The oil consumption is periodic. The length of one consumption cycle is 20s (second)
- The profile of consumption rate in one cycle is depicted by



The Consumption Rate

- The oil consumption is periodic. The length of one consumption cycle is 20s (second)
- The profile of consumption rate in one cycle is depicted by



The Pump

- The power of the pump is 2.2 l/s (liter/second)
- 2-second latency: if the pump is switched on (t_{2k+1}) or off (t_{2k+2}) at time points

$$0 \leq t_1 \leq t_2 \leq \cdots \leq t_i \leq t_{i+1} \leq \cdots,$$

  then

$$t_{i+1} - t_i \geq 2$$

  for any $i \geq 1$
- It is obvious that the pump can be turned on at most 5 times in one cycle

The Pump

- The power of the pump is 2.2 l/s (liter/second)
- 2-second latency: if the pump is switched on (t_{2k+1}) or off (t_{2k+2}) at time points

$$0 \leq t_1 \leq t_2 \leq \cdots \leq t_i \leq t_{i+1} \leq \cdots ,$$

  then

$$t_{i+1} - t_i \geq 2$$

  for any $i \geq 1$
- It is obvious that the pump can be turned on at most 5 times in one cycle

The Pump

- The power of the pump is 2.2 l/s (liter/second)
- 2-second latency: if the pump is switched on (t_{2k+1}) or off (t_{2k+2}) at time points

$$0 \leq t_1 \leq t_2 \leq \cdots \leq t_i \leq t_{i+1} \leq \cdots,$$

  then

$$t_{i+1} - t_i \geq 2$$

  for any $i \geq 1$
- It is obvious that the pump can be turned on at most 5 times in one cycle

Control Objectives

Determine the t_i's in order to

- $\mathrm{R_s}$ (*safety*): maintain

$$v(t) \in [V_{\min}, V_{\max}], \quad \forall t \in [0, \infty)$$

  - $v(t)$ denotes the oil volume in the accumulator at time t
  - $V_{\min} = 4.9l$ (liter)
  - $V_{\max} = 25.1l$

and considering the energy cost and wear of the system,

- $\mathrm{R_o}$ (*optimality*): minimize the average accumulated oil volume in the limit, i.e. minimize

$$\lim_{T \to \infty} \frac{1}{T} \int_{t=0}^{T} v(t) \mathrm{d}t$$

Control Objectives

Determine the t_i's in order to

- $\mathrm{R_s}$ (*safety*): maintain

$$v(t) \in [V_{\text{min}}, V_{\text{max}}], \quad \forall t \in [0, \infty)$$

  - $v(t)$ denotes the oil volume in the accumulator at time t
  - $V_{\text{min}} = 4.9l$ (liter)
  - $V_{\text{max}} = 25.1l$

and considering the energy cost and wear of the system,

- $\mathrm{R_o}$ (*optimality*): minimize the average accumulated oil volume in the limit, i.e. minimize

$$\lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} v(t)\mathrm{d}t$$

Control Objectives (Cont'd)

Both objectives should be achieved under constraints:

- R_{pl} (*pump latency*): $t_{i+1} - t_i \geq 2$
- R_{r} (*robustness*): uncertainties of the system should be taken into account:
  - fluctuation of consumption rate (if it is not 0), up to $f = 0.1 l/s$
  - imprecision in the measurement of oil volume, up to $\epsilon = 0.06 l$
  - imprecision in the measurement of time, up to $\delta = 0.015 s$.

Control Objectives (Cont'd)

Both objectives should be achieved under constraints:

- R_{pl} (*pump latency*): $t_{i+1} - t_i \geq 2$
- R_{r} (*robustness*): uncertainties of the system should be taken into account:
  - fluctuation of consumption rate (if it is not 0), up to $f = 0.1 l/s$
  - imprecision in the measurement of oil volume, up to $\epsilon = 0.06 l$
  - imprecision in the measurement of time, up to $\delta = 0.015 s$.

Control Objectives (Cont'd)

Both objectives should be achieved under constraints:

- R_{pl} (*pump latency*): $t_{i+1} - t_i \geq 2$
- R_{r} (*robustness*): uncertainties of the system should be taken into account:
  - fluctuation of consumption rate (if it is not 0), up to $f = 0.1l/s$
  - imprecision in the measurement of oil volume, up to $\epsilon = 0.06l$
  - imprecision in the measurement of time, up to $\delta = 0.015s$.

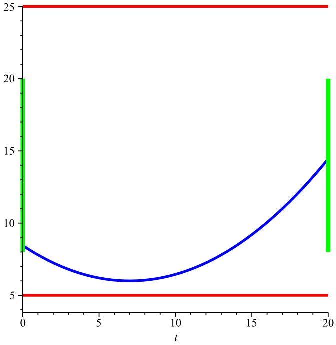
Control Objectives (Cont'd)

Both objectives should be achieved under constraints:

- R_{pl} (*pump latency*): $t_{i+1} - t_i \geq 2$
- R_{r} (*robustness*): uncertainties of the system should be taken into account:
  - fluctuation of consumption rate (if it is not 0), up to $f = 0.1 l/s$
  - imprecision in the measurement of oil volume, up to $\epsilon = 0.06 l$
  - imprecision in the measurement of time, up to $\delta = 0.015 s$.

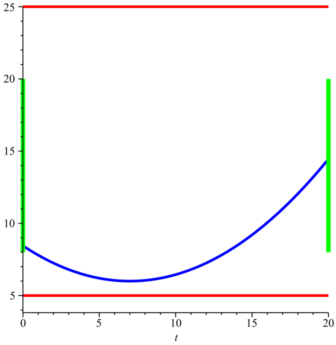
Localize the Controller

- $0 \leq t_1 \leq t_2 \leq \cdots \leq t_i \leq t_{i+1} \leq \cdots$
- Employing the periodicity
- Stable interval $[L, U] \subseteq [V_{\min}, V_{\max}]$



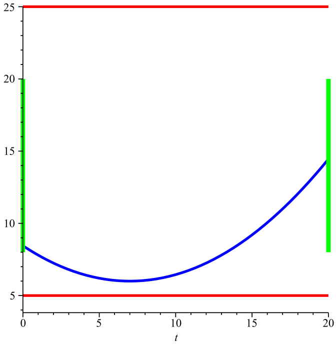
Localize the Controller

- $0 \leq t_1 \leq t_2 \leq \cdots \leq t_i \leq t_{i+1} \leq \cdots$
- Employing the periodicity
- Stable interval $[L, U] \subseteq [V_{\min}, V_{\max}]$

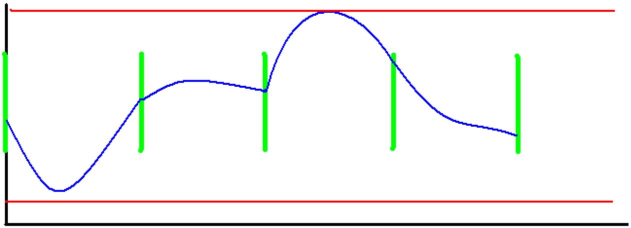


Localize the Controller

- $0 \leq t_1 \leq t_2 \leq \cdots \leq t_i \leq t_{i+1} \leq \cdots$
- Employing the periodicity
- Stable interval $[L, U] \subseteq [V_{\min}, V_{\max}]$

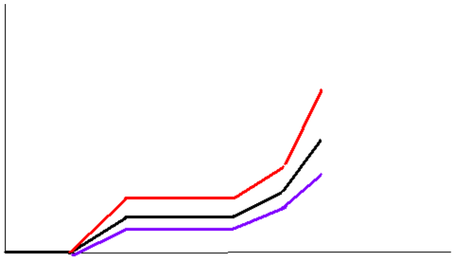


Repeated Cycles



Step 1: Modeling Oil Consumption

time	[2,4]	[8,10]	[10,12]	[14,16]	[16,18]
rate	1.2	1.2	2.5	1.7	0.5
- fluctuation of consumption rate: $f = 0.1$



Step 1: Modeling Oil Consumption

time	[2,4]	[8,10]	[10,12]	[14,16]	[16,18]
rate	1.2	1.2	2.5	1.7	0.5
- fluctuation of consumption rate: $f = 0.1$

$$
C_1 \mathrel{\widehat{=}}
\begin{array}{lcl}
(0\leq t\leq 2 & \longrightarrow & V_{out}=0) \\
\wedge\,(2\leq t\leq 4 & \longrightarrow & 1.1(t-2)\leq V_{out}\leq 1.3(t-2)) \\
\wedge\,(4\leq t\leq 8 & \longrightarrow & 2.2\leq V_{out}\leq 2.6) \\
\wedge\,(8\leq t\leq 10 & \longrightarrow & 2.2+1.1(t-8)\leq V_{out}\leq 2.6+1.3(t-8)) \\
\wedge\,(10\leq t\leq 12 & \longrightarrow & 4.4+2.4(t-10)\leq V_{out}\leq 5.2+2.6(t-10)) \\
\wedge\,(12\leq t\leq 14 & \longrightarrow & 9.2\leq V_{out}\leq 10.4) \\
\wedge\,(14\leq t\leq 16 & \longrightarrow & 9.2+1.6(t-14)\leq V_{out}\leq 10.4+1.8(t-14)) \\
\wedge\,(16\leq t\leq 18 & \longrightarrow & 12.4+0.4(t-16)\leq V_{out}\leq 14+0.6(t-16)) \\
\wedge\,(18\leq t\leq 20 & \longrightarrow & 13.2\leq V_{out}\leq 15.2)
\end{array}
$$

Step 1: Modeling the Pump

- We will first assume that the pump is activated at most twice in one cycle: t_1, t_2, t_3, t_4
- $t_{i+1} - t_i \geq 2$:

$$C_2 \;\widehat{=}\; \begin{array}{l} \quad(t_1{\geq}2 \wedge t_2{-}t_1{\geq}2 \wedge t_3{-}t_2{\geq}2 \wedge t_4{-}t_3{\geq}2 \wedge t_4{\leq}20) \\ \vee\,(t_1{\geq}2 \wedge t_2{-}t_1{\geq}2 \wedge t_2{\leq}20 \wedge t_3{=}20 \wedge t_4{=}20) \\ \vee\,(t_1{=}20 \wedge t_2{=}20 \wedge t_3{=}20 \wedge t_4{=}20) \end{array}.$$

- $2.2l/s$

$$C_3 \;\widehat{=}\; \begin{array}{lcl} \quad(0{\leq}t{\leq}t_1 & \longrightarrow & V_{in}{=}0) \\ \wedge\,(t_1{\leq}t{\leq}t_2 & \longrightarrow & V_{in}{=}2.2(t{-}t_1)) \\ \wedge\,(t_2{\leq}t{\leq}t_3 & \longrightarrow & V_{in}{=}2.2(t_2{-}t_1)) \\ \wedge\,(t_3{\leq}t{\leq}t_4 & \longrightarrow & V_{in}{=}2.2(t_2{-}t_1){+}2.2(t{-}t_3)) \\ \wedge\,(t_4{\leq}t{\leq}20 & \longrightarrow & V_{in}{=}2.2(t_2{+}t_4{-}t_1{-}t_3)) \end{array}.$$

Step 1: Modeling the Pump

- We will first assume that the pump is activated at most twice in one cycle: t_1, t_2, t_3, t_4
- $t_{i+1} - t_i \geq 2$:

$$C_2 \mathrel{\widehat{=}} \begin{array}{l} \quad (t_1{\geq}2 \wedge t_2{-}t_1{\geq}2 \wedge t_3{-}t_2{\geq}2 \wedge t_4{-}t_3{\geq}2 \wedge t_4{\leq}20) \\ \vee (t_1{\geq}2 \wedge t_2{-}t_1{\geq}2 \wedge t_2{\leq}20 \wedge t_3{=}20 \wedge t_4{=}20) \\ \vee (t_1{=}20 \wedge t_2{=}20 \wedge t_3{=}20 \wedge t_4{=}20) \end{array} .$$

- $2.2l/s$

$$C_3 \mathrel{\widehat{=}} \begin{array}{lcl} \quad (0{\leq}t{\leq}t_1 & \longrightarrow & V_{in}{=}0) \\ \wedge (t_1{\leq}t{\leq}t_2 & \longrightarrow & V_{in}{=}2.2(t{-}t_1)) \\ \wedge (t_2{\leq}t{\leq}t_3 & \longrightarrow & V_{in}{=}2.2(t_2{-}t_1)) \\ \wedge (t_3{\leq}t{\leq}t_4 & \longrightarrow & V_{in}{=}2.2(t_2{-}t_1){+}2.2(t{-}t_3)) \\ \wedge (t_4{\leq}t{\leq}20 & \longrightarrow & V_{in}{=}2.2(t_2{+}t_4{-}t_1{-}t_3)) \end{array} .$$

Step 1: Modeling the Pump

- We will first assume that the pump is activated at most twice in one cycle: t_1, t_2, t_3, t_4
- $t_{i+1} - t_i \geq 2$:

$$C_2 \widehat{=} \begin{array}{l} (t_1{\geq}2 \wedge t_2{-}t_1{\geq}2 \wedge t_3{-}t_2{\geq}2 \wedge t_4{-}t_3{\geq}2 \wedge t_4{\leq}20) \\ \vee (t_1{\geq}2 \wedge t_2{-}t_1{\geq}2 \wedge t_2{\leq}20 \wedge t_3{=}20 \wedge t_4{=}20) \\ \vee (t_1{=}20 \wedge t_2{=}20 \wedge t_3{=}20 \wedge t_4{=}20) \end{array}.$$

- $2.2l/s$

$$C_3 \widehat{=} \begin{array}{lcl} (0{\leq}t{\leq}t_1 & \longrightarrow & V_{in}{=}0) \\ \wedge (t_1{\leq}t{\leq}t_2 & \longrightarrow & V_{in}{=}2.2(t{-}t_1)) \\ \wedge (t_2{\leq}t{\leq}t_3 & \longrightarrow & V_{in}{=}2.2(t_2{-}t_1)) \\ \wedge (t_3{\leq}t{\leq}t_4 & \longrightarrow & V_{in}{=}2.2(t_2{-}t_1){+}2.2(t{-}t_3)) \\ \wedge (t_4{\leq}t{\leq}20 & \longrightarrow & V_{in}{=}2.2(t_2{+}t_4{-}t_1{-}t_3)) \end{array}.$$

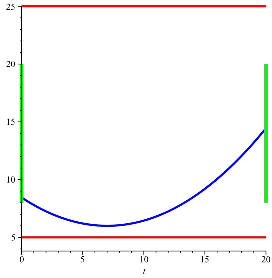
Step 1: Encoding Safety Requirements

- Oil volume in the accumulator:

$$C_4 \mathrel{\widehat{=}} v = v_0 + V_{in} - V_{out} \quad .$$

- Inductiveness and safety (considering robustness):

$$\begin{aligned} C_5 &\mathrel{\widehat{=}} t = 20 \longrightarrow L+0.2 \le v \le U-0.2 \\ C_6 &\mathrel{\widehat{=}} 0 \le t \le 20 \longrightarrow V_{\min}+0.2 \le v \le V_{\max}-0.2 \quad . \end{aligned}$$



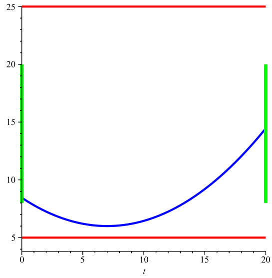
Step 1: Encoding Safety Requirements

- Oil volume in the accumulator:

$$C_4 \mathrel{\widehat{=}} v = v_0 + V_{in} - V_{out} \quad .$$

- Inductiveness and safety (considering robustness):

$$\begin{aligned} C_5 \quad &\widehat{=} \quad t = 20 \longrightarrow L+0.2 \leq v \leq U-0.2 \\ C_6 \quad &\widehat{=} \quad 0 \leq t \leq 20 \longrightarrow V_{\min}+0.2 \leq v \leq V_{\max}-0.2 \quad . \end{aligned}$$



Step 1: Encoding Safety Requirements (Cont'd)

$$\mathcal{S} \mathrel{\widehat{=}} \forall t, v, V_{in}, V_{out}.(C_1 \land C_3 \land C_4 \longrightarrow C_5 \land C_6)\,.$$

- C_1: oil consumed
- C_3: oil pumped
- C_4: oil in the accumulator
- C_5: inductiveness
- C_6: (local) safety

$$C_8 \mathrel{\widehat{=}} \forall v_0.\Big(C_7 \longrightarrow \exists t_1 t_2 t_3 t_4.(C_2 \land \mathcal{S})\Big)\,.$$

- $C_7 \mathrel{\widehat{=}} L \leq v_0 \leq U$
- C_2: 2-second latency

Step 1: Encoding Safety Requirements (Cont'd)

$$\mathcal{S} \mathrel{\widehat{=}} \forall t, v, V_{in}, V_{out}.(C_1 \wedge C_3 \wedge C_4 \longrightarrow C_5 \wedge C_6)\,.$$

- C_1: oil consumed
- C_3: oil pumped
- C_4: oil in the accumulator
- C_5: inductiveness
- C_6: (local) safety

$$C_8 \mathrel{\widehat{=}} \forall v_0.\left(C_7 \longrightarrow \exists t_1 t_2 t_3 t_4.(C_2 \wedge \mathcal{S})\right)\,.$$

- $C_7 \mathrel{\widehat{=}} L \leq v_0 \leq U$
- C_2: 2-second latency

Step 1: Encoding Safety Requirements (Cont'd)

$$\mathcal{S} \mathrel{\widehat{=}} \forall t, v, V_{in}, V_{out}.(C_1 \wedge C_3 \wedge C_4 \longrightarrow C_5 \wedge C_6)\,.$$

- C_1: oil consumed
- C_3: oil pumped
- C_4: oil in the accumulator
- C_5: inductiveness
- C_6: (local) safety

$$C_8 \mathrel{\widehat{=}} \forall v_0.\Big(C_7 \longrightarrow \exists t_1 t_2 t_3 t_4.\big(C_2 \wedge \mathcal{S}\big)\Big) \quad .$$

- $C_7 \mathrel{\widehat{=}} L \leq v_0 \leq U$
- C_2: 2-second latency

Deriving Constraints

Applying QE to

$$C_8 \mathrel{\widehat{=}} \forall v_0. \Big(C_7 \longrightarrow \exists t_1 t_2 t_3 t_4. \big(C_2 \wedge \mathcal{S} \big) \Big) \quad ,$$

we get

$$C_9 \mathrel{\widehat{=}} L \geq 5.1 \wedge U \leq 24.9 \wedge U - L \geq 2.4 \quad .$$

Deriving Constraints

Applying QE to

$$C_8 \mathrel{\widehat{=}} \forall v_0. \Big(C_7 \longrightarrow \exists t_1 t_2 t_3 t_4. \big(C_2 \wedge \mathcal{S} \big) \Big) \quad ,$$

we get

$$C_9 \mathrel{\widehat{=}} L \geq 5.1 \wedge U \leq 24.9 \wedge U - L \geq 2.4 \quad .$$

Deriving Constraints (Cont'd)

$$C_{10} \mathrel{\widehat{=}} C_2 \wedge C_7 \wedge C_9 \wedge \mathcal{S} \,.$$

- C_2: 2-second latency
- C_7 : $L \leq v_0 \leq U$
- C_9: constraint on L, U
- $\mathcal{S}$: safety and inductiveness

After QE:

$$\mathcal{D}(L, U, v_0, t_1, t_2, t_3, t_4) \mathrel{\widehat{=}} \bigvee_{i=1}^{92} D_i$$

Deriving Constraints (Cont'd)

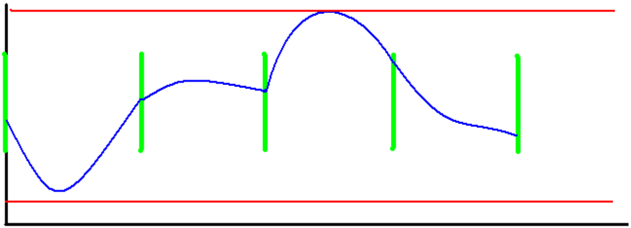
$$C_{10} \widehat{=} C_2 \wedge C_7 \wedge C_9 \wedge \mathcal{S}\,.$$

- C_2: 2-second latency
- C_7 : $L \leq v_0 \leq U$
- C_9: constraint on L, U
- $\mathcal{S}$: safety and inductiveness

After QE:

$$\mathcal{D}(L, U, v_0, t_1, t_2, t_3, t_4) \widehat{=} \bigvee_{i=1}^{92} D_i$$

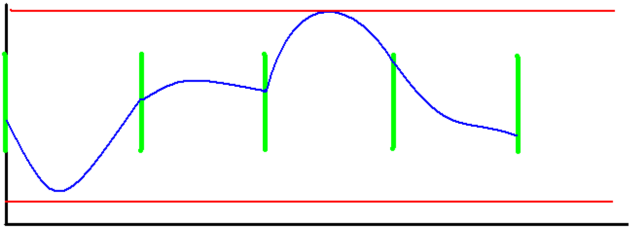
Step 2: Optimization Criterion



$\mathrm{R_o}$ (*optimality*): minimize the average accumulated oil volume in the limit, i.e. minimize

$$\lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} v(t)\mathrm{d}t$$

Optimization Criterion (Contd.)



- R'_{o} : $\min_{[L,U]} \max_{v_0 \in [L,U]} \min_{\mathbf{t}} \frac{1}{20} \int_{t=0}^{20} v(t)\mathrm{d}t$.

Step 2: Encoding the Optimization Criterion

Cost function:

$$g(v_0, t_1, t_2, t_3, t_4) \;\widehat{=}\; \frac{1}{20}\int_{t=0}^{20} v(t)\mathrm{d}t$$
$$= \frac{20v_0 + 1.1(t_1^2 - t_2^2 + t_3^2 - t_4^2 - 40t_1 + 40t_2 - 40t_3 + 40t_4) - 132.2}{20}$$

R'_{o} can be encoded into

$$\exists L, U.\Big(C_9 \wedge \forall v_0.\big(C_7 \longrightarrow \exists t_1 t_2 t_3 t_4.(\mathcal{D} \wedge g \leq z)\big)\Big),$$

which is equivalent to $z \geq z^*$ or $z > z^*$

Step 2: Encoding the Optimization Criterion

Cost function:

$$\begin{aligned} g(v_0, t_1, t_2, t_3, t_4) &\mathrel{\widehat{=}} \frac{1}{20}\int_{t=0}^{20} v(t)\mathrm{d}t \\ &= \frac{20v_0+1.1(t_1^2-t_2^2+t_3^2-t_4^2-40t_1+40t_2-40t_3+40t_4)-132.2}{20} \end{aligned}$$

R'_{o} can be encoded into

$$\exists L, U.\Big(C_9 \wedge \forall v_0.\big(C_7 \longrightarrow \exists t_1 t_2 t_3 t_4.(\mathcal{D} \wedge g \leq z)\big)\Big),$$

which is equivalent to $z \geq z^*$ or $z > z^*$

Step 3: Performing QE

$$\exists L, U. \Big(C_9 \wedge \forall v_0. \big(C_7 \longrightarrow \exists t_1 t_2 t_3 t_4. (\mathcal{D} \wedge g \leq z) \big) \Big)$$

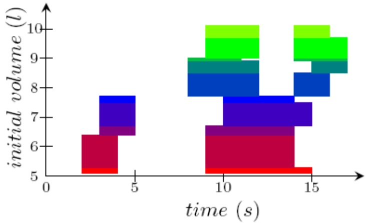
- the inner $\exists$: *qudratic programming*
- the outer $\exists$: discretization

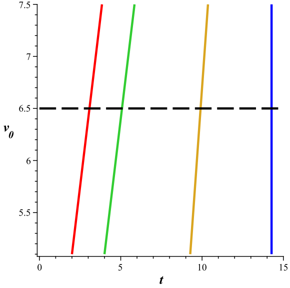
$$L \geq 5.1 \wedge U \leq 24.9 \wedge U - L \geq 2.4$$

- the middle $\forall$: divide and conquer

Optimal Controllers with 2 Activations

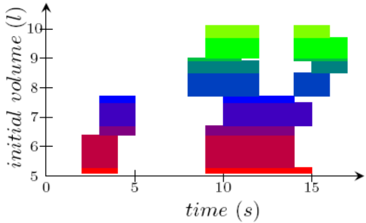
- In [Cassez et al hscc09], the optimal value 7.95 is obtained at interval [5.1,8.3]
- Using our approach, the optimal value is 7.53 (a 5% improvement) and the corresponding interval is $[5.1, 7.5]$
- Comparison of local optimal controllers: (the left one comes from [Cassez et al hscc09])

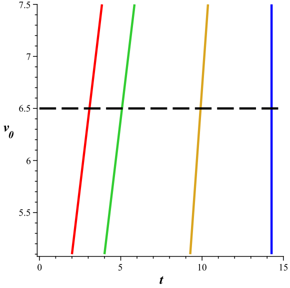




Optimal Controllers with 2 Activations

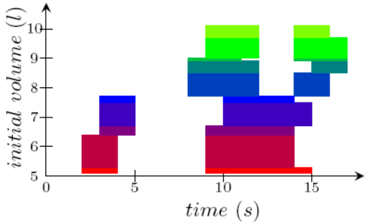
- In [Cassez et al hscc09], the optimal value 7.95 is obtained at interval [5.1,8.3]
- Using our approach, the optimal value is 7.53 (a 5% improvement) and the corresponding interval is $[5.1, 7.5]$
- Comparison of local optimal controllers: (the left one comes from [Cassez et al hscc09])

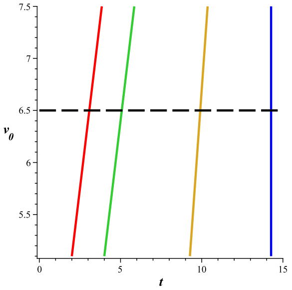




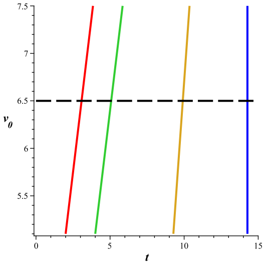
Optimal Controllers with 2 Activations

- In [Cassez et al hscc09], the optimal value 7.95 is obtained at interval [5.1,8.3]
- Using our approach, the optimal value is 7.53 (a 5% improvement) and the corresponding interval is $[5.1, 7.5]$
- Comparison of local optimal controllers: (the left one comes from [Cassez et al hscc09])





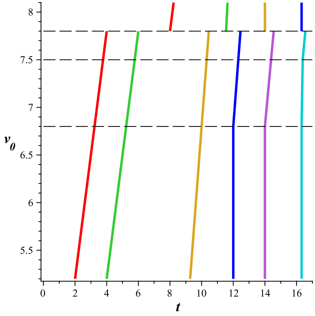
Local Optimal Controllers — 2 Activations



$$t_1 = \frac{10v_0 - 25}{13} \wedge t_2 = \frac{10v_0 + 1}{13} \wedge t_3 = \frac{10v_0 + 153}{22} \wedge t_4 = \frac{157}{11}$$

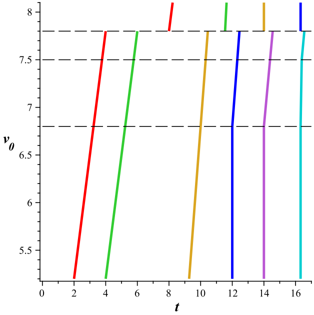
Improvement by Increasing Activations

- The pump is allowed to be switched on at most 3 times in one cycle
- The optimal average accumulated oil volume 7.35 (a 7.5% improvement) is obtained at interval $[5.2, 8.1]$
- The local optimal controllers corresponding to $v_0 \in [5.2, 8.1]$:



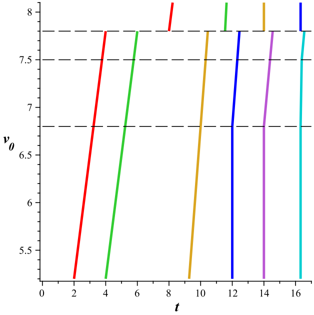
Improvement by Increasing Activations

- The pump is allowed to be switched on at most 3 times in one cycle
- The optimal average accumulated oil volume 7.35 (a 7.5% improvement) is obtained at interval $[5.2, 8.1]$
- The local optimal controllers corresponding to $v_0 \in [5.2, 8.1]$:

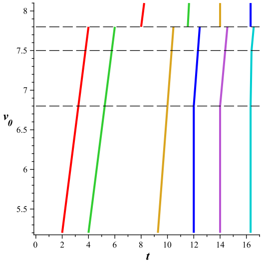


Improvement by Increasing Activations

- The pump is allowed to be switched on at most 3 times in one cycle
- The optimal average accumulated oil volume 7.35 (a 7.5% improvement) is obtained at interval $[5.2, 8.1]$
- The local optimal controllers corresponding to $v_0 \in [5.2, 8.1]$:



Local Optimal Controllers — 3 Activations



$$
\begin{cases}
t_1=\frac{10v_0-26}{13} \wedge t_2=\frac{10v_0}{13} \wedge t_3=\frac{5v_0+76}{11} \wedge t_4=12 \wedge t_5=14 \wedge t_6=\frac{359}{22} & v_0\in[5.2,6.8) \\
t_1=\frac{10v_0-26}{13} \wedge t_2=\frac{10v_0}{13} \wedge t_3=\frac{5v_0+76}{11} \wedge t_4=\frac{5v_0+98}{11} \wedge t_5=\frac{5v_0+92}{9} \wedge t_6=\frac{20v_0+3095}{198} & v_0\in[6.8,7.5) \\
t_1=\frac{10v_0-26}{13} \wedge t_2=\frac{10v_0}{13} \wedge t_3=\frac{5v_0+76}{11} \wedge t_4=\frac{5v_0+98}{11} \wedge t_5=\frac{5v_0+92}{9} \wedge t_6=\frac{5v_0+110}{9} & v_0\in[7.5,7.8) \\
t_1=\frac{10v_0+26}{13} \wedge t_2=\frac{45v_0+1300}{143} \wedge t_3=14 \wedge t_4=\frac{359}{22} \wedge t_5=20 \wedge t_6=20 & v_0\in[7.8,8.1]
\end{cases}
$$

Three Activations are Enough

Proposition

For each admissible $[L, U]$, each $v_0 \in [L, U]$, and any local control strategy s_4 with at least 4 activations subject to R_{lu}, R_i and R_{ls}, there exists a local control strategy s_3 subject to R_{lu}, R_i and R_{ls} with 3 activations such that

$$\frac{1}{20}\int_{t=0}^{20} v_{s_3}(t)\mathrm{d}t < \frac{1}{20}\int_{t=0}^{20} v_{s_4}(t)\mathrm{d}t$$

where $v_{s_3}(t)$ (resp. $v_{s_4}(t)$) is the oil volume in the accumulator at t with s_3 (resp. s_4).

Outline

Related Work

Hybrid CSP (HCSP) due to He [He 1994], Zhou et al [Zhou et al, 1995], is an extension of CSP by introducing **differential equation** to describe continuous evolution and three kinds of **interruptions** to model the interaction between continuous evolution and discrete jumps.

1. J. He: From CSP to Hybrid Systems: A Classical Mind, Prentice Hall 1994
2. C. Zhou, J. Wang and A.P. Ravn: Formal Description of Hybrid Systems, LNCS 1066
3. C. Zhou, A.P. Ravn, M.R. Hansen: Extended Duration Calculus, LNCS 736
4. J. Liu, J. Lv, Z. Quan, N. Zhan, H. Zhao, C. Zhou and L. Zou: A calculus for HCSP. LNCS 6461.
5. S. Wang, N. Zhan and D. Guelev: An assume/guarantee based compositional calculus for HCSP. LNCS 7287.
6. N. Zhan, S. Wang and D. Guelev: Extending Hoare logic to hybrid systems. Technical Report ISCAS-SKLCS-13-02.

Interruptions of HCSP

- **Communication events**: message passing $ch!m$ and $ch?x$
- **Communication interruption:**

$$P \trianglerighteq (ch?x \rightarrow Q)$$

  initially proceeds like P, and is interrupted by communication along ch, and then proceeds like Q.
- **Example:** $\langle \dot{s} = v,\ \dot{v} = a\rangle \trianglerighteq (ch_{r2t}?x \rightarrow (x = eb \rightarrow EB))$
- **Timeout events (Timeout interruption)** $P \trianglerighteq_t Q$ behaves as P for up to t time units, and it continues with Q after t time units.
- **Example:** $\langle \dot{s} = v,\ \dot{v} = a\rangle \trianglerighteq_T \langle \dot{s} = v,\ \dot{v} = -b\rangle$
- **Boolean events (Boundary interruption):** $\langle F(\dot{s}, s) = 0 \& B\rangle$ means that the process behaves like $F(\dot{s}, s) = 0$ subject to B holds, but will be interrupted whenever B is violated.
- **Example:** $\langle \dot{s} = v,\ \dot{v} = a \& v < v_{ebi}\rangle; EB$

Syntax of HCSP

- Denote by $\mathcal{V}$ ranged over $x, y, s, \ldots$ the set of variables, and by Σ ranged over $ch, ch_1, \ldots$ the set of channels.
- The syntax of HCSP:

$$
\begin{array}{lll}
P & ::= & \textbf{skip} \mid x := e \mid \mathrm{wait}\ d \mid ch?x \mid ch!e \mid P; Q \mid B \to P \mid P \sqcup Q \\
& & \mid P^* \mid \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \mid \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \unrhd_d Q \\
& & \mid \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \unrhd []_{i \in I} (ch_i * \to Q_i) \\
S & ::= & P \mid S \| S
\end{array}
$$

Here $ch, ch_i \in \Sigma$, ch_i* stands for a communication event, i.e., either $ch_i?x$ or $ch_i!e$, $x, s \in \mathcal{V}$, B and e are Boolean and arithmetic expressions, d is a non-negative real constant, P, Q, Q_i are sequential processes, and S stands for a system, i.e., an HCSP process.

Informal Meaning

- $\langle F(\dot{s}, s) = 0 \& B\rangle$ defines a dynamical evolution by the ODE. B is a first order formula of s, which defines a domain in the sense that, if s is beyond B, the statement terminates; otherwise it goes forward.
- $\langle F(\dot{s}, s) = 0 \& B\rangle \trianglerighteq_d P$ behaves like $\langle F(\dot{s}, s) = 0 \& B\rangle$ if it can terminate within d time units. Otherwise, after d (inclusive) time units, it will behave like P.
- $\langle F(\dot{s}, s) = 0 \& B\rangle \trianglerighteq [\!]_{i \in I}(io_i \to P_i)$ behaves like $\langle F(\dot{s}, s) = 0 \& B\rangle$ until a communication in the following context appears. Then it behaves like P_i immediately after communication io_i occurs.
- $P \sqcup Q$ is the ***internal choice*** of CSP.
- P^* means P can be repeated arbitrarily finitely many times.
- $P \parallel Q$ behaves as if P and Q are executed independently except that all communications along the common channels shared by P and Q are to be synchronized.
- **Note that shared variable is not allowed in HCSP**, so in $P \parallel Q$, P and Q cannot have shared variables, and neither shared input nor output channels.

Informal Meaning

- $\langle F(\dot{s}, s) = 0 \& B \rangle$ defines a dynamical evolution by the ODE. B is a first order formula of s, which defines a domain in the sense that, if s is beyond B, the statement terminates; otherwise it goes forward.
- $\langle F(\dot{s}, s) = 0 \& B \rangle \unrhd_d P$ behaves like $\langle F(\dot{s}, s) = 0 \& B \rangle$ if it can terminate within d time units. Otherwise, after d (inclusive) time units, it will behave like P.
- $\langle F(\dot{s}, s) = 0 \& B \rangle \unrhd [\!]_{i \in I}(io_i \rightarrow P_i)$ behaves like $\langle F(\dot{s}, s) = 0 \& B \rangle$ until a communication in the following context appears. Then it behaves like P_i immediately after communication io_i occurs.
- $P \sqcup Q$ is the ***internal choice*** of CSP.
- P^* means P can be repeated arbitrarily finitely many times.
- $P \| Q$ behaves as if P and Q are executed independently except that all communications along the common channels shared by P and Q are to be synchronized.
- **Note that shared variable is not allowed in HCSP**, so in $P \| Q$, P and Q cannot have shared variables, and neither shared input nor output channels.

Informal Meaning

- $\langle F(\dot{s}, s) = 0 \& B \rangle$ defines a dynamical evolution by the ODE. B is a first order formula of s, which defines a domain in the sense that, if s is beyond B, the statement terminates; otherwise it goes forward.
- $\langle F(\dot{s}, s) = 0 \& B \rangle \trianglerighteq_d P$ behaves like $\langle F(\dot{s}, s) = 0 \& B \rangle$ if it can terminate within d time units. Otherwise, after d (inclusive) time units, it will behave like P.
- $\langle F(\dot{s}, s) = 0 \& B \rangle \trianglerighteq [\![_{i \in I}(io_i \to P_i)$ behaves like $\langle F(\dot{s}, s) = 0 \& B \rangle$ until a communication in the following context appears. Then it behaves like P_i immediately after communication io_i occurs.
- $P \sqcup Q$ is the ***internal choice*** of CSP.
- P^* means P can be repeated arbitrarily finitely many times.
- $P \parallel Q$ behaves as if P and Q are executed independently except that all communications along the common channels shared by P and Q are to be synchronized.
- **Note that shared variable is not allowed in HCSP**, so in $P \parallel Q$, P and Q cannot have shared variables, and neither shared input nor output channels.

Informal Meaning

- $\langle F(\dot{s}, s) = 0 \& B \rangle$ defines a dynamical evolution by the ODE. B is a first order formula of s, which defines a domain in the sense that, if s is beyond B, the statement terminates; otherwise it goes forward.
- $\langle F(\dot{s}, s) = 0 \& B \rangle \unrhd_d P$ behaves like $\langle F(\dot{s}, s) = 0 \& B \rangle$ if it can terminate within d time units. Otherwise, after d (inclusive) time units, it will behave like P.
- $\langle F(\dot{s}, s) = 0 \& B \rangle \unrhd [\!]_{i \in I}(io_i \to P_i)$ behaves like $\langle F(\dot{s}, s) = 0 \& B \rangle$ until a communication in the following context appears. Then it behaves like P_i immediately after communication io_i occurs.
- $P \sqcup Q$ is the ***internal choice*** of CSP.
- P^* means P can be repeated arbitrarily finitely many times.
- $P \| Q$ behaves as if P and Q are executed independently except that all communications along the common channels shared by P and Q are to be synchronized.
- **Note that shared variable is not allowed in HCSP**, so in $P \| Q$, P and Q cannot have shared variables, and neither shared input nor output channels.

Informal Meaning

- $\langle F(\dot{s}, s) = 0\&B\rangle$ defines a dynamical evolution by the ODE. B is a first order formula of s, which defines a domain in the sense that, if s is beyond B, the statement terminates; otherwise it goes forward.
- $\langle F(\dot{s}, s) = 0\&B\rangle \unrhd_d P$ behaves like $\langle F(\dot{s}, s) = 0\&B\rangle$ if it can terminate within d time units. Otherwise, after d (inclusive) time units, it will behave like P.
- $\langle F(\dot{s}, s) = 0\&B\rangle \unrhd [\!]_{i\in I}(io_i \to P_i)$ behaves like $\langle F(\dot{s}, s) = 0\&B\rangle$ until a communication in the following context appears. Then it behaves like P_i immediately after communication io_i occurs.
- $P \sqcup Q$ is the ***internal choice*** of CSP.
- P^* means P can be repeated arbitrarily finitely many times.
- $P \parallel Q$ behaves as if P and Q are executed independently except that all communications along the common channels shared by P and Q are to be synchronized.
- Note that shared variable is not allowed in HCSP, so in $P \parallel Q$, P and Q cannot have shared variables, and neither shared input nor output channels.

Informal Meaning

- $\langle F(\dot{s}, s) = 0 \& B \rangle$ defines a dynamical evolution by the ODE. B is a first order formula of s, which defines a domain in the sense that, if s is beyond B, the statement terminates; otherwise it goes forward.
- $\langle F(\dot{s}, s) = 0 \& B \rangle \trianglerighteq_d P$ behaves like $\langle F(\dot{s}, s) = 0 \& B \rangle$ if it can terminate within d time units. Otherwise, after d (inclusive) time units, it will behave like P.
- $\langle F(\dot{s}, s) = 0 \& B \rangle \trianglerighteq [\!]_{i \in I}(io_i \rightarrow P_i)$ behaves like $\langle F(\dot{s}, s) = 0 \& B \rangle$ until a communication in the following context appears. Then it behaves like P_i immediately after communication io_i occurs.
- $P \sqcup Q$ is the ***internal choice*** of CSP.
- P^* means P can be repeated arbitrarily finitely many times.
- $P \| Q$ behaves as if P and Q are executed independently except that all communications along the common channels shared by P and Q are to be synchronized.
- **Note that shared variable is not allowed in HCSP**, so in $P \| Q$, P and Q cannot have shared variables, and neither shared input nor output channels.

Derived Operators

Note that some primitives of CSP, timed CSP and even some of the constructs of HCSP are derivable, e.g.,

- **stop**: $\textbf{stop} \stackrel{\text{def}}{=} t := 0; \langle \dot{t} = 1 \& \mathit{true} \rangle$.
- **wait**: $\text{wait } d \stackrel{\text{def}}{=} t := 0; \langle \dot{t} = 1 \& t < d \rangle$.
- **external choice**: $[]_{i \in I}(ch_i * \to Q_i) \stackrel{\text{def}}{=} \textbf{stop} \trianglerighteq []_{i \in I}(ch_i * \to Q_i)$.
- **timeout**:

$$\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq_d Q \stackrel{\text{def}}{=} \begin{array}{l} t := 0; \\ \langle F(\dot{s}, s) = 0 \wedge \dot{t} = 1 \& t < d \wedge B \rangle; \\ t \geq d \to Q \end{array} .$$

Some Examples

1. Plant Control (PLC)

$$
\begin{array}{ll}
 & (\langle F(s, \dot{s}, u) = 0 \rangle \trianglerighteq c_{p2c}!s \rightarrow c_{c2p}?u)^* \\
\| & (\text{wait } d; c_{p2c}?v; c_{c2p}!contl(v))^*
\end{array}
$$

2. MA (simplified): A train is moving until it reaches Emergency Brake condition (B_{eb}), and takes deceleration ($-b$) to return to safe region ($\neg B_{eb}$). During moving, it periodically receives from RBC new MA or emergency brake message, and updates B_{eb} or decelerates with $-b$ accordingly.

$$
\begin{array}{ll}
 & \left(\begin{array}{ll} & \langle (\dot{s} = v, \dot{v} = a) \& \neg B_{eb}(x) \rangle \\ \trianglerighteq & (c_{r2t}?x \rightarrow (B_{eb}(x) \vee x = eb) \rightarrow a := -b) \end{array} \right)^* \\
\| & (\textbf{wait } T; (c_{r2t}!ma \sqcup c_{r2t}!eb))^*
\end{array}
$$

Some Examples

1. Plant Control (*PLC*)

$$
\begin{array}{ll}
 & (\langle F(s, \dot{s}, u) = 0 \rangle \trianglerighteq c_{p2c}!s \rightarrow c_{c2p}?u)^* \\
\| & (\text{wait } d; c_{p2c}?v; c_{c2p}!contl(v))^*
\end{array}
$$

2. MA (simplified): A train is moving until it reaches Emergency Brake condition (B_{eb}), and takes deceleration ($-b$) to return to safe region ($\neg B_{eb}$). During moving, it periodically receives from RBC new MA or emergency brake message, and updates B_{eb} or decelerates with $-b$ accordingly.

$$
\begin{array}{ll}
 & \left(\begin{array}{ll} & \langle (\dot{s} = v, \dot{v} = a) \& \neg B_{eb}(x) \rangle \\ \trianglerighteq & (c_{r2t}?x \rightarrow (B_{eb}(x) \vee x = eb) \rightarrow a := -b) \end{array} \right)^* \\
\| & (\textbf{wait } T; (c_{r2t}!ma \sqcup c_{r2t}!eb))^*
\end{array}
$$

Outline

Notations

- A ***timed communication*** is of the form $\langle ch.c, b\rangle$, where $ch \in \Sigma$, $c \in \mathbb{R}$ and $b \in \mathbb{R}^+$. $T\Sigma$ denotes the set of all timed communications.
- The set of all ***timed traces*** is $T\Sigma^*_{\leq} = \{\gamma \in T\Sigma^* \mid \text{if } \langle ch_1.c_1, b_1\rangle \text{ precedes } \langle ch_2.c_2, b_2\rangle, \text{then } b_1 \leq b_2\}$.
- We introduce a global clock now over $\mathbb{R}^+$ as a system variable to record the time in the execution of a process, and two system variables, rdy and tr, to represent the ***ready set*** of communication events and the ***timed communication trace*** accumulated.
- A state σ of P is an assignment to associate a value from the respective domain to each variable in $\mathcal{V}^+(P)$, where $\mathcal{V}^+(P) = \mathcal{V}(P) \cup \{rdy, tr, now\}$.
- Given two states σ_1 and σ_2, we say σ_1 and σ_2 are **parallelable** iff $Dom(\sigma_1) \cap Dom(\sigma_2) = \{rdy, tr, now\}$ and $\sigma_1(now) = \sigma_2(now)$.
- A ***flow***, ranging over H, H_1, defined on a time interval, assigns a state to each point in the interval.
- Each transition relation has the form of $(P, \sigma) \xrightarrow{\alpha} (P', \sigma', H)$.

Notations

- A ***timed communication*** is of the form $\langle ch.c, b\rangle$, where $ch \in \Sigma$, $c \in \mathbb{R}$ and $b \in \mathbb{R}^+$. $T\Sigma$ denotes the set of all timed communications.
- The set of all ***timed traces*** is
  $T\Sigma^*_{\leq} = \{\gamma \in T\Sigma^* \mid \text{if } \langle ch_1.c_1, b_1\rangle \text{ precedes } \langle ch_2.c_2, b_2\rangle, \text{then } b_1 \leq b_2\}$.
- We introduce a global clock now over $\mathbb{R}^+$ as a system variable to record the time in the execution of a process, and two system variables, rdy and tr, to represent the ***ready set*** of communication events and the ***timed communication trace*** accumulated.
- A state σ of P is an assignment to associate a value from the respective domain to each variable in $\mathcal{V}^+(P)$, where $\mathcal{V}^+(P) = \mathcal{V}(P) \cup \{rdy, tr, now\}$.
- Given two states σ_1 and σ_2, we say σ_1 and σ_2 are **parallelable** iff $Dom(\sigma_1) \cap Dom(\sigma_2) = \{rdy, tr, now\}$ and $\sigma_1(now) = \sigma_2(now)$.
- A ***flow***, ranging over H, H_1, defined on a time interval, assigns a state to each point in the interval.
- Each transition relation has the form of $(P, \sigma) \xrightarrow{\alpha} (P', \sigma', H)$.

Notations

- A ***timed communication*** is of the form $\langle ch.c, b\rangle$, where $ch \in \Sigma$, $c \in \mathbb{R}$ and $b \in \mathbb{R}^+$. $T\Sigma$ denotes the set of all timed communications.
- The set of all ***timed traces*** is $T\Sigma^*_{\leq} = \{\gamma \in T\Sigma^* \mid \text{if } \langle ch_1.c_1, b_1\rangle \text{ precedes } \langle ch_2.c_2, b_2\rangle, \text{then } b_1 \leq b_2\}$.
- We introduce a global clock *now* over $\mathbb{R}^+$ as a system variable to record the time in the execution of a process, and two system variables, *rdy* and *tr*, to represent the ***ready set*** of communication events and the ***timed communication trace*** accumulated.
- A state σ of P is an assignment to associate a value from the respective domain to each variable in $\mathcal{V}^+(P)$, where $\mathcal{V}^+(P) = \mathcal{V}(P) \cup \{rdy, tr, now\}$.
- Given two states σ_1 and σ_2, we say σ_1 and σ_2 are **parallelable** iff $Dom(\sigma_1) \cap Dom(\sigma_2) = \{rdy, tr, now\}$ and $\sigma_1(now) = \sigma_2(now)$.
- A ***flow***, ranging over H, H_1, defined on a time interval, assigns a state to each point in the interval.
- Each transition relation has the form of $(P, \sigma) \xrightarrow{\alpha} (P', \sigma', H)$.

Notations

- A ***timed communication*** is of the form $\langle ch.c, b\rangle$, where $ch \in \Sigma$, $c \in \mathbb{R}$ and $b \in \mathbb{R}^+$. $T\Sigma$ denotes the set of all timed communications.
- The set of all ***timed traces*** is $T\Sigma^*_{\leq} = \{\gamma \in T\Sigma^* \mid \text{if } \langle ch_1.c_1, b_1\rangle \text{ precedes } \langle ch_2.c_2, b_2\rangle, \text{then } b_1 \leq b_2\}$.
- We introduce a global clock *now* over $\mathbb{R}^+$ as a system variable to record the time in the execution of a process, and two system variables, *rdy* and *tr*, to represent the ***ready set*** of communication events and the ***timed communication trace*** accumulated.
- A state σ of P is an assignment to associate a value from the respective domain to each variable in $\mathcal{V}^+(P)$, where $\mathcal{V}^+(P) = \mathcal{V}(P) \cup \{rdy, tr, now\}$.
- Given two states σ_1 and σ_2, we say σ_1 and σ_2 are **parallelable** iff $Dom(\sigma_1) \cap Dom(\sigma_2) = \{rdy, tr, now\}$ and $\sigma_1(now) = \sigma_2(now)$.
- A ***flow***, ranging over H, H_1, defined on a time interval, assigns a state to each point in the interval.
- Each transition relation has the form of $(P, \sigma) \xrightarrow{\alpha} (P', \sigma', H)$.

Notations

- A ***timed communication*** is of the form $\langle ch.c, b\rangle$, where $ch \in \Sigma$, $c \in \mathbb{R}$ and $b \in \mathbb{R}^+$. $T\Sigma$ denotes the set of all timed communications.
- The set of all ***timed traces*** is $T\Sigma^*_{\leq} = \{\gamma \in T\Sigma^* \mid \text{if } \langle ch_1.c_1, b_1\rangle \text{ precedes } \langle ch_2.c_2, b_2\rangle, \text{then } b_1 \leq b_2\}$.
- We introduce a global clock *now* over $\mathbb{R}^+$ as a system variable to record the time in the execution of a process, and two system variables, *rdy* and *tr*, to represent the ***ready set*** of communication events and the ***timed communication trace*** accumulated.
- A state σ of P is an assignment to associate a value from the respective domain to each variable in $\mathcal{V}^+(P)$, where $\mathcal{V}^+(P) = \mathcal{V}(P) \cup \{rdy, tr, now\}$.
- Given two states σ_1 and σ_2, we say σ_1 and σ_2 are **parallelable** iff $Dom(\sigma_1) \cap Dom(\sigma_2) = \{rdy, tr, now\}$ and $\sigma_1(now) = \sigma_2(now)$.
- A ***flow***, ranging over H, H_1, defined on a time interval, assigns a state to each point in the interval.
- Each transition relation has the form of $(P, \sigma) \xrightarrow{\alpha} (P', \sigma', H)$.

Notations

- A ***timed communication*** is of the form $\langle ch.c, b\rangle$, where $ch \in \Sigma$, $c \in \mathbb{R}$ and $b \in \mathbb{R}^+$. $T\Sigma$ denotes the set of all timed communications.
- The set of all ***timed traces*** is $T\Sigma^*_{\leq} = \{\gamma \in T\Sigma^* \mid \text{if } \langle ch_1.c_1, b_1\rangle \text{ precedes } \langle ch_2.c_2, b_2\rangle, \text{then } b_1 \leq b_2\}$.
- We introduce a global clock *now* over $\mathbb{R}^+$ as a system variable to record the time in the execution of a process, and two system variables, *rdy* and *tr*, to represent the ***ready set*** of communication events and the ***timed communication trace*** accumulated.
- A state σ of P is an assignment to associate a value from the respective domain to each variable in $\mathcal{V}^+(P)$, where $\mathcal{V}^+(P) = \mathcal{V}(P) \cup \{rdy, tr, now\}$.
- Given two states σ_1 and σ_2, we say σ_1 and σ_2 are **parallelable** iff $Dom(\sigma_1) \cap Dom(\sigma_2) = \{rdy, tr, now\}$ and $\sigma_1(now) = \sigma_2(now)$.
- A ***flow***, ranging over H, H_1, defined on a time interval, assigns a state to each point in the interval.
- Each transition relation has the form of $(P, \sigma) \xrightarrow{\alpha} (P', \sigma', H)$.

Notations

- A ***timed communication*** is of the form $\langle ch.c, b\rangle$, where $ch \in \Sigma$, $c \in \mathbb{R}$ and $b \in \mathbb{R}^+$. $T\Sigma$ denotes the set of all timed communications.
- The set of all ***timed traces*** is $T\Sigma^*_{\leq} = \{\gamma \in T\Sigma^* \mid \text{if } \langle ch_1.c_1, b_1\rangle \text{ precedes } \langle ch_2.c_2, b_2\rangle, \text{then } b_1 \leq b_2\}$.
- We introduce a global clock now over $\mathbb{R}^+$ as a system variable to record the time in the execution of a process, and two system variables, rdy and tr, to represent the ***ready set*** of communication events and the ***timed communication trace*** accumulated.
- A state σ of P is an assignment to associate a value from the respective domain to each variable in $\mathcal{V}^+(P)$, where $\mathcal{V}^+(P) = \mathcal{V}(P) \cup \{rdy, tr, now\}$.
- Given two states σ_1 and σ_2, we say σ_1 and σ_2 are **parallelable** iff $Dom(\sigma_1) \cap Dom(\sigma_2) = \{rdy, tr, now\}$ and $\sigma_1(now) = \sigma_2(now)$.
- A ***flow***, ranging over H, H_1, defined on a time interval, assigns a state to each point in the interval.
- Each transition relation has the form of $(P, \sigma) \xrightarrow{\alpha} (P', \sigma', H)$.

Transition Rules

$$(\mathbf{skip}, \sigma) \xrightarrow{\tau} (\epsilon, \sigma[tr + \tau]) \quad \text{(Skip)}$$

$$(\epsilon, \sigma) \xrightarrow{d} (\epsilon, \sigma[now \mapsto \sigma(now) + d], H_d) \quad \text{(Idle)}$$

$$(x := e, \sigma) \xrightarrow{\tau} (\epsilon, \sigma[x \mapsto \sigma(e), tr \mapsto \sigma(tr) \cdot \langle \tau, \sigma(now) \rangle]) \quad \text{(Ass)}$$

$$\frac{\sigma(tr).ch? \notin \sigma(rdy)}{(ch?x, \sigma) \xrightarrow{\tau} (ch?x, \sigma[rdy \mapsto \sigma(rdy) \cup \{\sigma(tr).ch?\}])} \quad \text{(In-1)}$$

$$\frac{\sigma(tr).ch? \in \sigma(rdy)}{(ch?x, \sigma) \xrightarrow{d} (ch?x, \sigma[now \mapsto \sigma(now) + d], H_d)} \quad \text{(In-2)}$$

$$\frac{\sigma(tr).ch? \in \sigma(rdy)}{(ch?x, \sigma) \xrightarrow{ch?b} (\epsilon, \ \sigma[x \mapsto b, tr + ch.b, rdy \mapsto \sigma(rdy) \backslash \{\sigma(tr).ch?\}] \ )} \quad \text{(In-3)}$$

Transition Rules (Cont'd)

$$\frac{\sigma(tr).ch! \notin \sigma(rdy)}{(ch!e, \sigma) \xrightarrow{\tau} (ch!e, \sigma[rdy \mapsto \sigma(rdy) \cup \{\sigma(tr).ch!\}])} \quad \text{(Out-1)}$$

$$\frac{\sigma(tr).ch! \in \sigma(rdy)}{(ch!e, \sigma) \xrightarrow{d} (ch!e, \sigma[now \mapsto \sigma(now) + d], H_d)} \quad \text{(Out-2)}$$

$$\frac{\sigma(tr).ch! \in \sigma(rdy)}{(ch!e, \sigma) \xrightarrow{ch!\sigma(e)} (\epsilon, \ \sigma[tr + ch.\sigma(e), rdy \mapsto \sigma(rdy) \backslash \{\sigma(tr).ch!\}] \ )} \quad \text{(Out-3)}$$

$$\frac{\begin{array}{l} S(t) \text{ is a trajectory of } \mathcal{F}(\dot{s}, s) = 0 \text{ s.t. } (S(0) = \sigma(s) \\ \wedge \forall t \in [0, d].(\mathcal{F}(\dot{S}(t), S(t)) = 0 \wedge \sigma(B[s \mapsto S(t)]) = true)) \end{array}}{(\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle, \sigma) \xrightarrow{d} \left(\begin{array}{l} \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle, \\ \sigma[now \mapsto \sigma(now) + d, s \mapsto S(d)], H_{d,s} \end{array} \right)} \quad \text{(Cont-1)}$$

$$\frac{\begin{array}{l} (\sigma(B) = false) \text{ or } (S(t) \text{ is a trajectory of } \mathcal{F}(\dot{s}, s) = 0 \text{ s.t.} \\ \exists \varepsilon > 0.(S(0) = \sigma(s) \\ \wedge \forall t \in (0, \varepsilon].(\mathcal{F}(\dot{S}(t), S(t)) = 0 \wedge \sigma(B[s \mapsto S(t)]) = false))) \end{array}}{(\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle, \sigma) \xrightarrow{\tau} (\epsilon, \sigma[s \mapsto \lim_{t \to 0} S(t), tr \mapsto \sigma(tr) \cdot \langle \tau, \sigma(now) \rangle])} \quad \text{(Cont-2)}$$

Transition Rules (Cont'd)

$$\frac{\begin{array}{c}(ch_i*; Q_i, \sigma) \xrightarrow{d} (ch_i*; Q_i, \sigma'_i, H_i), \quad \forall i \in I \\ (\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle, \sigma) \xrightarrow{d} (\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle, \sigma', H)\end{array}}{\begin{array}{l}(\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle \trianglerighteq []_{i\in I}(ch_i* \to Q_i), \sigma) \xrightarrow{d} \\ \qquad \left(\begin{array}{l}\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle \trianglerighteq []_{i\in I}(ch_i* \to Q_i), \\ \sigma'[rdy \mapsto \cup_{i\in I}\sigma'_i(rdy)], \\ H[rdy \mapsto \cup_{i\in I}\sigma'_i(rdy)]\end{array}\right)\end{array}} \quad \text{(IntP-1)}$$

$$\frac{(ch_j*; Q_j, \sigma) \xrightarrow{ch_j*} (Q_j, \sigma'), \exists j \in I}{(\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle \trianglerighteq []_{i\in I}(ch_i* \to Q_i), \sigma) \xrightarrow{ch_j*} (Q_j, \sigma')} \quad \text{(IntP-2)}$$

$$\frac{(\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle, \sigma) \xrightarrow{\tau} (\epsilon, \sigma'))}{(\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle \trianglerighteq []_{i\in I}(ch_i* \to Q_i), \sigma) \xrightarrow{\tau} (\epsilon, \sigma')} \quad \text{(IntP-3)}$$

Transition Rules (Cont'd)

$$\frac{(P_1,\sigma_1) \xrightarrow{d} (P_1',\sigma_1',H_1),\quad (P_2,\sigma_2) \xrightarrow{d} (P_2',\sigma_2',H_2),\quad \forall ch \in \Sigma(P_1)\cap\Sigma(P_2).\neg((P_1,\sigma_1\uplus\sigma_2) \xrightarrow{ch*} \wedge (P_2,\sigma_1\uplus\sigma_2) \xrightarrow{\overline{ch*}})}{(P_1 \parallel P_2,\sigma_1\uplus\sigma_2) \xrightarrow{d} (P_1' \parallel P_2',(\sigma_1'\uplus\sigma_2'),H_1\uplus H_2)} \quad \text{(Par-1)}$$

$$\frac{(P_1,\sigma_1) \xrightarrow{\beta} (P_1',\sigma_1'),\quad \Sigma(\beta)\notin\Sigma(P_1)\cap\Sigma(P_2)}{(P_1 \parallel P_2,\sigma_1\uplus\sigma_2) \xrightarrow{\beta} (P_1' \parallel P_2,\sigma_1'\uplus\sigma_2)} \quad \text{(Par-2)}$$

$$\frac{(P_1,\sigma_1) \xrightarrow{ch*} (P_1',\sigma_1'),\quad (P_2,\sigma_2) \xrightarrow{\overline{ch*}} (P_2',\sigma_2'),}{(P_1 \parallel P_2,\sigma_1\uplus\sigma_2) \xrightarrow{comm(ch*,\overline{ch*})} (P_1' \parallel P_2',\sigma_1'\uplus\sigma_2')} \quad \text{(Par-3)}$$

$$(\epsilon \parallel \epsilon,\sigma_1\uplus\sigma_2) \xrightarrow{\tau} (\epsilon,\sigma_1\uplus\sigma_2) \quad \text{(Par-4)}$$

Transition Rules (Cont'd)

$$\frac{\sigma(B) = true}{(B \to P, \sigma) \xrightarrow{\tau} (P, \sigma[tr + \tau])} \text{ (Cond-1)} \qquad \frac{\sigma(B) = false}{(B \to P, \sigma) \xrightarrow{\tau} (\epsilon, \sigma[tr + \tau])} \text{ (Cond-2)}$$

$$\frac{(P, \sigma) \xrightarrow{\alpha} (P', \sigma', H) \quad P' \neq \epsilon}{(P; Q, \sigma) \xrightarrow{\alpha} (P'; Q, \sigma', H)} \text{ (Seq-1)} \qquad \frac{(P, \sigma) \xrightarrow{\alpha} (\epsilon, \sigma', H)}{(P; Q, \sigma) \xrightarrow{\alpha} (Q, \sigma', H)} \text{ (Seq-2)}$$

$$(P \sqcup Q, \sigma) \xrightarrow{\tau} (P, \sigma[tr + \tau]) \text{ (IntC-1)} \qquad (P \sqcup Q, \sigma) \xrightarrow{\tau} (Q, \sigma[tr + \tau]) \text{ (IntC-2)}$$

$$\frac{(P, \sigma) \xrightarrow{\alpha} (P', \sigma', H) \quad P' \neq \epsilon}{(P^*, \sigma) \xrightarrow{\alpha} (P'; P^*, \sigma', H)} \text{ (Rep-1)} \qquad \frac{(P, \sigma) \xrightarrow{\alpha} (\epsilon, \sigma', H)}{(P^*, \sigma) \xrightarrow{\alpha} (P^*, \sigma', H)} \text{ (Rep-2)}$$

$$(P^*, \sigma) \xrightarrow{\tau} (\epsilon, \sigma[tr + \tau]) \text{ (Rep-3)}$$

Definitions

- **Super-dense computation**: two time granularities: **macro time** for environment, and **micro time** for computation, and **micro time** will be abstracted to be 0 at abstract level.
- Given two flows H_1 and H_2 defined on $[r_1, r_2]$ and $[r_2, r_3]$ respectively, their ***concatenation*** $H_1^\frown H_2$ is the flow defined on $[r_1, r_3]$ such that $H_1^\frown H_2(t)$ is equal to $H_1(t)$ if $t \in [r_1, r_2)$, and $H_2(t)$ if $t \in [r_2, r_3)$.
- Given a process P and a state σ_0, if there is a sequence of transitions:

$$(P, \sigma_0) \xrightarrow{\alpha_0} (P_1, \sigma_1, H_1)$$
$$\ldots$$
$$(P_{n-1}, \sigma_{n-1}) \xrightarrow{\alpha_{n-1}} (P_n, \sigma_n, H_n)$$

  then we define $H_1^\frown \ldots ^\frown H_n$ as a ***flow*** from P_1 to P_n starting from σ_0.

- The sequence $B_1^\frown \ldots ^\frown B_n$ as a ***behavior*** from P_1 to P_n starting from σ_0, where B_i is H_i if H_i is not empty, empty otherwise if H_i is empty but H_{i+1} is not, σ_i otherwise.
- When P_n is ϵ, we will call them ***complete flow*** and ***complete behavior*** of P with respect to σ_0 respectively.

Definitions

- **Super-dense computation**: two time granularities: **macro time** for environment, and **micro time** for computation, and **micro time** will be abstracted to be 0 at abstract level.
- Given two flows H_1 and H_2 defined on $[r_1, r_2]$ and $[r_2, r_3]$ respectively, their ***concatenation*** $H_1^\frown H_2$ is the flow defined on $[r_1, r_3]$ such that $H_1^\frown H_2(t)$ is equal to $H_1(t)$ if $t \in [r_1, r_2)$, and $H_2(t)$ if $t \in [r_2, r_3)$.
- Given a process P and a state σ_0, if there is a sequence of transitions:

$$
\begin{array}{l}
(P, \sigma_0) \xrightarrow{\alpha_0} (P_1, \sigma_1, H_1) \\
\ldots \\
(P_{n-1}, \sigma_{n-1}) \xrightarrow{\alpha_{n-1}} (P_n, \sigma_n, H_n)
\end{array}
$$

then we define $H_1^\frown \ldots ^\frown H_n$ as a ***flow*** from P_1 to P_n starting from σ_0.

- The sequence $B_1^\frown \ldots ^\frown B_n$ as a ***behavior*** from P_1 to P_n starting from σ_0, where B_i is H_i if H_i is not empty, empty otherwise if H_i is empty but H_{i+1} is not, σ_i otherwise.
- When P_n is ϵ, we will call them ***complete flow*** and ***complete behavior*** of P with respect to σ_0 respectively.

Definitions

- **Super-dense computation**: two time granularities: **macro time** for environment, and **micro time** for computation, and **micro time** will be abstracted to be 0 at abstract level.
- Given two flows H_1 and H_2 defined on $[r_1, r_2]$ and $[r_2, r_3]$ respectively, their ***concatenation*** $H_1^\frown H_2$ is the flow defined on $[r_1, r_3]$ such that $H_1^\frown H_2(t)$ is equal to $H_1(t)$ if $t \in [r_1, r_2)$, and $H_2(t)$ if $t \in [r_2, r_3)$.
- Given a process P and a state σ_0, if there is a sequence of transitions:

$$
\begin{array}{l}
(P, \sigma_0) \xrightarrow{\alpha_0} (P_1, \sigma_1, H_1) \\
\ldots \\
(P_{n-1}, \sigma_{n-1}) \xrightarrow{\alpha_{n-1}} (P_n, \sigma_n, H_n)
\end{array}
$$

  then we define $H_1^\frown \ldots ^\frown H_n$ as a ***flow*** from P_1 to P_n starting from σ_0.
- The sequence $B_1^\frown \ldots ^\frown B_n$ as a ***behavior*** from P_1 to P_n starting from σ_0, where B_i is H_i if H_i is not empty, empty otherwise if H_i is empty but H_{i+1} is not, σ_i otherwise.
- When P_n is ϵ, we will call them ***complete flow*** and ***complete behavior*** of P with respect to σ_0 respectively.

Definitions

- **Super-dense computation**: two time granularities: **macro time** for environment, and **micro time** for computation, and **micro time** will be abstracted to be 0 at abstract level.
- Given two flows H_1 and H_2 defined on $[r_1, r_2]$ and $[r_2, r_3]$ respectively, their ***concatenation*** $H_1^\frown H_2$ is the flow defined on $[r_1, r_3]$ such that $H_1^\frown H_2(t)$ is equal to $H_1(t)$ if $t \in [r_1, r_2)$, and $H_2(t)$ if $t \in [r_2, r_3)$.
- Given a process P and a state σ_0, if there is a sequence of transitions:

$$
\begin{array}{l}
(P, \sigma_0) \xrightarrow{\alpha_0} (P_1, \sigma_1, H_1) \\
\ldots \\
(P_{n-1}, \sigma_{n-1}) \xrightarrow{\alpha_{n-1}} (P_n, \sigma_n, H_n)
\end{array}
$$

  then we define $H_1^\frown \ldots ^\frown H_n$ as a ***flow*** from P_1 to P_n starting from σ_0.
- The sequence $B_1^\frown \ldots ^\frown B_n$ as a ***behavior*** from P_1 to P_n starting from σ_0, where B_i is H_i if H_i is not empty, empty otherwise if H_i is empty but H_{i+1} is not, σ_i otherwise.
- When P_n is ϵ, we will call them ***complete flow*** and ***complete behavior*** of P with respect to σ_0 respectively.

Outline

Introduction

- **Hybrid Hoare Logic** (HHL) was first proposed in [Zhou et al 2010], which is an extension of **Hoare logic** to hybrid system, used to specify and reason about hybrid systems modelled by HCSP.
- The assertion logic of HHL consists of two parts: **the first-order logic** and **Duration Calculus (DC)**.
- **FOL** is used to specify discrete properties, represented by ***pre-*** and ***post-condition***, while **DC** is used to specify continuous evolution.
- In **HHL**, a hybrid system is modelled by **HCSP**.
- The proof system of **HHL** consists of the following three parts:
  - **axioms and inference rules for FOL**,
  - **axioms and inference rules for DC**, and
  - **axioms and inference rules for the constructs of HCSP**.
- A theorem prover based on **Isabelle/HOL** has been implemented, and applied to model and verify **CTCS-3** [Zou et al 2013a].
- In [Wang, Zhan& Guelev 2012] and [Zhan, Wang & Guelev 2013], compositional proof system for HHL was investigated.

Introduction

- **Hybrid Hoare Logic** (HHL) was first proposed in [Zhou et al 2010], which is an extension of **Hoare logic** to hybrid system, used to specify and reason about hybrid systems modelled by HCSP.
- The assertion logic of HHL consists of two parts: **the first-order logic** and **Duration Calculus (DC)**.
- **FOL** is used to specify discrete properties, represented by ***pre-*** and ***post-condition***, while **DC** is used to specify continuous evolution.
- In **HHL**, a hybrid system is modelled by **HCSP**.
- The proof system of **HHL** consists of the following three parts:
    - **axioms and inference rules for FOL**,
    - **axioms and inference rules for DC**, and
    - **axioms and inference rules for the constructs of HCSP**.
- A theorem prover based on **Isabelle/HOL** has been implemented, and applied to model and verify **CTCS-3** [Zou et al 2013a].
- In [Wang, Zhan& Guelev 2012] and [Zhan, Wang & Guelev 2013], compositional proof system for HHL was investigated.

Introduction

- **Hybrid Hoare Logic** (HHL) was first proposed in [Zhou et al 2010], which is an extension of **Hoare logic** to hybrid system, used to specify and reason about hybrid systems modelled by HCSP.
- The assertion logic of HHL consists of two parts: **the first-order logic** and **Duration Calculus (DC)**.
- **FOL** is used to specify discrete properties, represented by ***pre-*** and ***post-condition***, while **DC** is used to specify continuous evolution.
- In **HHL**, a hybrid system is modelled by **HCSP**.
- The proof system of **HHL** consists of the following three parts:
  - **axioms and inference rules for FOL**,
  - **axioms and inference rules for DC**, and
  - **axioms and inference rules for the constructs of HCSP**.
- A theorem prover based on Isabelle/HOL has been implemented, and applied to model and verify **CTCS-3** [Zou et al 2013a].
- In [Wang, Zhan& Guelev 2012] and [Zhan, Wang & Guelev 2013], compositional proof system for HHL was investigated.

Introduction

- **Hybrid Hoare Logic** (HHL) was first proposed in [Zhou et al 2010], which is an extension of **Hoare logic** to hybrid system, used to specify and reason about hybrid systems modelled by HCSP.
- The assertion logic of HHL consists of two parts: **the first-order logic** and **Duration Calculus (DC)**.
- **FOL** is used to specify discrete properties, represented by ***pre-*** and ***post-condition***, while **DC** is used to specify continuous evolution.
- In **HHL**, a hybrid system is modelled by **HCSP**.
- The proof system of **HHL** consists of the following three parts:
  - **axioms and inference rules for FOL**,
  - **axioms and inference rules for DC**, and
  - **axioms and inference rules for the constructs of HCSP**.
- A theorem prover based on **Isabelle/HOL** has been implemented, and applied to model and verify **CTCS-3** [Zou et al 2013a].
- In [Wang, Zhan& Guelev 2012] and [Zhan, Wang & Guelev 2013], compositional proof system for HHL was investigated.

Introduction

- **Hybrid Hoare Logic** (HHL) was first proposed in [Zhou et al 2010], which is an extension of **Hoare logic** to hybrid system, used to specify and reason about hybrid systems modelled by HCSP.
- The assertion logic of HHL consists of two parts: **the first-order logic** and **Duration Calculus (DC)**.
- **FOL** is used to specify discrete properties, represented by ***pre-*** and ***post-condition***, while **DC** is used to specify continuous evolution.
- In **HHL**, a hybrid system is modelled by **HCSP**.
- The proof system of **HHL** consists of the following three parts:
  - **axioms and inference rules for FOL**,
  - **axioms and inference rules for DC**, and
  - **axioms and inference rules for the constructs of HCSP**.
- A theorem prover based on Isabelle/HOL has been implemented, and applied to model and verify **CTCS-3** [Zou et al 2013a].
- In [Wang, Zhan& Guelev 2012] and [Zhan, Wang & Guelev 2013], compositional proof system for HHL was investigated.

Introduction

- **Hybrid Hoare Logic** (HHL) was first proposed in [Zhou et al 2010], which is an extension of **Hoare logic** to hybrid system, used to specify and reason about hybrid systems modelled by HCSP.
- The assertion logic of HHL consists of two parts: **the first-order logic** and **Duration Calculus (DC)**.
- **FOL** is used to specify discrete properties, represented by ***pre-*** and ***post-condition***, while **DC** is used to specify continuous evolution.
- In **HHL**, a hybrid system is modelled by **HCSP**.
- The proof system of **HHL** consists of the following three parts:
  - **axioms and inference rules for FOL**,
  - **axioms and inference rules for DC**, and
  - **axioms and inference rules for the constructs of HCSP**.
- A theorem prover based on **Isabelle/HOL** has been implemented, and applied to model and verify **CTCS-3** [Zou et al 2013a].
- In [Wang, Zhan& Guelev 2012] and [Zhan, Wang & Guelev 2013], compositional proof system for HHL was investigated.

Introduction

- **Hybrid Hoare Logic** (HHL) was first proposed in [Zhou et al 2010], which is an extension of **Hoare logic** to hybrid system, used to specify and reason about hybrid systems modelled by HCSP.
- The assertion logic of HHL consists of two parts: **the first-order logic** and **Duration Calculus (DC)**.
- **FOL** is used to specify discrete properties, represented by ***pre-*** and ***post-condition***, while **DC** is used to specify continuous evolution.
- In **HHL**, a hybrid system is modelled by **HCSP**.
- The proof system of **HHL** consists of the following three parts:
  - **axioms and inference rules for FOL**,
  - **axioms and inference rules for DC**, and
  - **axioms and inference rules for the constructs of HCSP**.
- A theorem prover based on **Isabelle/HOL** has been implemented, and applied to model and verify **CTCS-3** [Zou et al 2013a].
- In [Wang, Zhan& Guelev 2012] and [Zhan, Wang & Guelev 2013], compositional proof system for HHL was investigated.

History Formulas

State Expression

- **Syntax**:

$$S \quad ::= \quad 1 \mid 0 \mid R(e_1, \ldots, e_n) \mid \neg S \mid S_1 \vee S_2$$

where $R(e_1, \ldots, e_n)$ is a n-ary predicate over expressions $e_1, \ldots, e_n$, normally of the form $p(x_1, \ldots, x_n) \rhd 0$ with $\rhd \in \{\geq, >, =, \neq, \leq, <\}$ and $p(x_1, \ldots, x_n)$ are polynomials.

- **Semantics**: Given a state σ, a state expression S is interpreted as

$$\begin{array}{rcl} \sigma(1) & = & 1 \\ \sigma(0) & = & 0 \\ \sigma(R(e_1, \ldots, e_n)) & = & \begin{cases} 1, & \text{if } R(\sigma(e_1), \ldots, \sigma(e_n)); \\ 0, & \text{otherwise} \end{cases} \\ \sigma(\neg S) & = & 1 - \sigma(S) \\ \sigma(S_1 \vee S_2) & = & \max\{\sigma(S_1), \sigma(S_2)\} \end{array}$$

History Formulas

State Expression

- **Syntax**:

$$S \quad ::= \quad 1 \mid 0 \mid R(e_1, \ldots, e_n) \mid \neg S \mid S_1 \vee S_2$$

where $R(e_1, \ldots, e_n)$ is a n-ary predicate over expressions $e_1, \ldots, e_n$, normally of the form $p(x_1, \ldots, x_n) \rhd 0$ with $\rhd \in \{\geq, >, =, \neq, \leq, <\}$ and $p(x_1, \ldots, x_n)$ are polynomials.

- **Semantics**: Given a state σ, a state expression S is interpreted as

$$\begin{array}{rcl}
\sigma(1) & = & 1 \\
\sigma(0) & = & 0 \\
\sigma(R(e_1, \ldots, e_n)) & = & \begin{cases} 1, & \text{if } R(\sigma(e_1), \ldots, \sigma(e_n)); \\ 0, & \text{otherwise} \end{cases} \\
\sigma(\neg S) & = & 1 - \sigma(S) \\
\sigma(S_1 \vee S_2) & = & \max\{\sigma(S_1), \sigma(S_2)\}
\end{array}$$

History Formulas (Cont'd)

History Formulas

- **Syntax**:

$$HF \quad ::= \quad \ell < T \mid \ell = T \mid \ell > T \mid \lceil S \rceil^0 \mid \neg HF \mid HF_1 \frown HF_2 \mid HF_2 \vee HF_2$$

- **Semantics**: Given a flow H and a reference interval $[a, b]$ with $a, b \in \mathrm{Dom}(H)$, and $a \leq b$, the meaning of HF is defined as:
  - $H, [b, e] \models \ell \rhd T$ iff $e - b \rhd T$, where $\rhd \in \{\leq, >, =, \neq, \leq, <\}$;
  - $H, [b, e] \models \lceil S \rceil^0$ iff $b = e$, and $H(b)(S) = 1$;
  - $H, [b, e] \models \neg HF$ iff $H, [b, e] \not\models HF$;
  - $H, [b, e] \models HF_1 \wedge HF_2$ iff $H, [b, e] \models HF_1$ and $H, [b, e] \models HF_2$;
  - $H, [b, e] \models HF_1 \vee HF_2$ iff $H, [b, e] \models HF_1$ or $H, [b, e] \models HF_2$;
  - $H, [b, e] \models HF_1 \frown HF_2$ iff there is $m \in [b, e]$ s.t. $H, [b, m] \models HF_1$ and $H, [m, e] \models HF_2$.

History Formulas (Cont'd)

History Formulas

- **Syntax**:

$$HF \quad ::= \quad \ell < T \mid \ell = T \mid \ell > T \mid \lceil S \rceil^0 \mid \neg HF \mid HF_1^\frown HF_2 \mid HF_2 \vee HF_2$$

- **Semantics**: Given a flow H and a reference interval $[a, b]$ with $a, b \in \mathrm{Dom}(H)$, and $a \leq b$, the meaning of HF is defined as:
  - $H, [b, e] \models \ell \triangleright T$ iff $e - b \triangleright T$, where $\triangleright \in \{\leq, >, =, \neq, \leq, <\}$;
  - $H, [b, e] \models \lceil S \rceil^0$ iff $b = e$, and $H(b)(S) = 1$;
  - $H, [b, e] \models \neg HF$ iff $H, [b, e] \not\models HF$;
  - $H, [b, e] \models HF_1 \wedge HF_2$ iff $H, [b, e] \models HF_1$ and $H, [b, e] \models HF_2$;
  - $H, [b, e] \models HF_1 \vee HF_2$ iff $H, [b, e] \models HF_1$ or $H, [b, e] \models HF_2$;
  - $H, [b, e] \models HF_1^\frown HF_2$ iff there is $m \in [b, e]$ s.t. $H, [b, m] \models HF_1$ and $H, [m, e] \models HF_2$.

History Formulas (Cont'd)

Internal of History Formula

$HF^<$ means that HF holds on the interval derived from the referred interval by excluding its endpoint, defined by:

$$
\begin{array}{lcl}
(\ell < T)^< & \stackrel{\mathrm{def}}{=} & (\ell < T) \\
(\ell = T)^< & \stackrel{\mathrm{def}}{=} & (\ell = T) \\
(\ell > T)^< & \stackrel{\mathrm{def}}{=} & \ell > T \\
(\lceil S \rceil^0)^< & \stackrel{\mathrm{def}}{=} & \ell = 0 \\
\lceil S \rceil^< & \stackrel{\mathrm{def}}{=} & \lceil S \rceil \\
(HF_1 ^\frown HF_2)^< & \stackrel{\mathrm{def}}{=} & (HF_1)^< {}^\frown (HF_2)^< \\
(HF_1 \wedge HF_2)^< & \stackrel{\mathrm{def}}{=} & (HF_1)^< \wedge (HF_2)^< \\
(HF_1 \vee HF_2)^< & \stackrel{\mathrm{def}}{=} & (HF_1)^< \vee (HF_2)^<
\end{array}
$$

Hoare Assertion

Hoare Assertion

A Hoare assertion of HHL is of the form

$$\{A\}P\{R; HF\}$$

- P is a HCSP process;
- **Precondition** A specifies values of $\mathcal{V}(P)$ before an execution of P;
- **Postcondition** R specifies values of $\mathcal{V}(P)$ when P terminates;
- HF is a history formula to describe the execution history of P.

Semantics

We say a Hoare assertion $\{A\}P\{R; HF\}$ is *valid*, denoted by $\models \{A\}P\{R; HF\}$, iff for any initial state σ_1, if $(P, \sigma_1) \xrightarrow{\alpha^*} (\epsilon, \sigma_2, H)$ then $\sigma_1 \models A$ implies $\sigma_2 \models R$ and $H, [\sigma_1(now), \sigma_2(now)] \models HF$.

Hoare Assertion

Hoare Assertion

A Hoare assertion of HHL is of the form

$$\{A\}P\{R; HF\}$$

- P is a HCSP process;
- **Precondition** A specifies values of $\mathcal{V}(P)$ before an execution of P;
- **Postcondition** R specifies values of $\mathcal{V}(P)$ when P terminates;
- HF is a history formula to describe the execution history of P.

Semantics

We say a Hoare assertion $\{A\}P\{R; HF\}$ is *valid*, denoted by $\models \{A\}P\{R; HF\}$, iff for any initial state σ_1, if $(P, \sigma_1) \xrightarrow{\alpha^*} (\epsilon, \sigma_2, H)$ then $\sigma_1 \models A$ implies $\sigma_2 \models R$ and $H, [\sigma_1(now), \sigma_2(now)] \models HF$.

Hoare Assertion

Hoare Assertion

A Hoare assertion of HHL is of the form

$$\{A\}P\{R; HF\}$$

- P is a HCSP process;
- **Precondition** A specifies values of $\mathcal{V}(P)$ before an execution of P;
- **Postcondition** R specifies values of $\mathcal{V}(P)$ when P terminates;
- HF is a history formula to describe the execution history of P.

Semantics

We say a Hoare assertion $\{A\}P\{R; HF\}$ is *valid*, denoted by $\models \{A\}P\{R; HF\}$, iff for any initial state σ_1, if $(P, \sigma_1) \xrightarrow{\alpha^*} (\epsilon, \sigma_2, H)$ then $\sigma_1 \models A$ implies $\sigma_2 \models R$ and $H, [\sigma_1(now), \sigma_2(now)] \models HF$.

Hoare Assertion

Hoare Assertion

A Hoare assertion of HHL is of the form

$$\{A\}P\{R; HF\}$$

- P is a HCSP process;
- **Precondition** A specifies values of $\mathcal{V}(P)$ before an execution of P;
- **Postcondition** R specifies values of $\mathcal{V}(P)$ when P terminates;
- HF is a history formula to describe the execution history of P.

Semantics

We say a Hoare assertion $\{A\}P\{R; HF\}$ is *valid*, denoted by $\models \{A\}P\{R; HF\}$, iff for any initial state σ_1, if $(P, \sigma_1) \xrightarrow{\alpha^*} (\epsilon, \sigma_2, H)$ then $\sigma_1 \models A$ implies $\sigma_2 \models R$ and $H, [\sigma_1(now), \sigma_2(now)] \models HF$.

Hoare Assertion

Hoare Assertion

A Hoare assertion of HHL is of the form

$$\{A\}P\{R; HF\}$$

- P is a HCSP process;
- **Precondition** A specifies values of $\mathcal{V}(P)$ before an execution of P;
- **Postcondition** R specifies values of $\mathcal{V}(P)$ when P terminates;
- HF is a history formula to describe the execution history of P.

Semantics

We say a Hoare assertion $\{A\}P\{R; HF\}$ is *valid*, denoted by $\models \{A\}P\{R; HF\}$, iff for any initial state σ_1, if $(P, \sigma_1) \xrightarrow{\alpha^*} (\epsilon, \sigma_2, H)$ then $\sigma_1 \models A$ implies $\sigma_2 \models R$ and $H, [\sigma_1(now), \sigma_2(now)] \models HF$.

Hoare Assertion

Hoare Assertion

A Hoare assertion of HHL is of the form

$$\{A\}P\{R; HF\}$$

- P is a HCSP process;
- **Precondition** A specifies values of $\mathcal{V}(P)$ before an execution of P;
- **Postcondition** R specifies values of $\mathcal{V}(P)$ when P terminates;
- HF is a history formula to describe the execution history of P.

Semantics

We say a Hoare assertion $\{A\}P\{R; HF\}$ is *valid*, denoted by $\models \{A\}P\{R; HF\}$, iff for any initial state σ_1, if $(P, \sigma_1) \xrightarrow{\alpha^*} (\epsilon, \sigma_2, H)$ then $\sigma_1 \models A$ implies $\sigma_2 \models R$ and $H, [\sigma_1(now), \sigma_2(now)] \models HF$.

Hoare Assertion

Hoare Assertion

A Hoare assertion of HHL is of the form

$$\{A\}P\{R; HF\}$$

- P is a HCSP process;
- **Precondition** A specifies values of $\mathcal{V}(P)$ before an execution of P;
- **Postcondition** R specifies values of $\mathcal{V}(P)$ when P terminates;
- HF is a history formula to describe the execution history of P.

Semantics

We say a Hoare assertion $\{A\}P\{R; HF\}$ is *valid*, denoted by $\models \{A\}P\{R; HF\}$, iff for any initial state σ_1, if $(P, \sigma_1) \xrightarrow{\alpha^*} (\epsilon, \sigma_2, H)$ then $\sigma_1 \models A$ implies $\sigma_2 \models R$ and $H, [\sigma_1(now), \sigma_2(now)] \models HF$.

Hoare Assertion (Cont'd)

Remarks

- For a parallel process $P_1 \parallel \ldots \parallel P_n$, the assertion becomes

$$\{A_1, \ldots, A_n\} P_1 \parallel \ldots \parallel P_n \{R_1, \ldots, R_n; HF_1, \ldots, HF_n\}$$

  where A_i, R_i, HF_i are (first order or DC) formulas of $\mathcal{V}(P_i)$ ($i = 1, \ldots, n$) separately. The validity can be defined similarly.
- Note that we can essentially put A and R as parts of history formula HF like the form $\lceil A \rceil^0 {}^\frown HF {}^\frown \lceil R \rceil^0$.

Example

In **PLC**, we can specify the system as:

$$\{s = s_0 \wedge u = u_0 \wedge Ctrl(u_0, s_0), A_2\}\ PLC$$
$$\{R_1, R_2; (l = T)^\frown \lceil | s - s_{targ} | \leq \epsilon \rceil, HF_2\}$$

where $Ctrl(u, s)$ may express a controllable property, and the other formulas are not elaborated here.

Hoare Assertion (Cont'd)

Remarks

- For a parallel process $P_1 \parallel \ldots \parallel P_n$, the assertion becomes

$$\{A_1, \ldots, A_n\} P_1 \parallel \ldots \parallel P_n \{R_1, \ldots, R_n; HF_1, \ldots, HF_n\}$$

  where A_i, R_i, HF_i are (first order or DC) formulas of $\mathcal{V}(P_i)$ ($i = 1, \ldots, n$) separately. The validity can be defined similarly.
- Note that we can essentially put A and R as parts of history formula HF like the form $\lceil A \rceil^0 \frown HF \frown \lceil R \rceil^0$.

Example

In **PLC**, we can specify the system as:

$$\{s = s_0 \wedge u = u_0 \wedge Ctrl(u_0, s_0), A_2\}\ PLC$$
$$\{R_1, R_2; (I = T) \frown \lceil | s - s_{targ} | \leq \epsilon \rceil, HF_2\}$$

where $Ctrl(u, s)$ may express a controllable property, and the other formulas are not elaborated here.

Hoare Assertion (Cont'd)

Remarks

- For a parallel process $P_1 \parallel \ldots \parallel P_n$, the assertion becomes

$$\{A_1, \ldots, A_n\} P_1 \parallel \ldots \parallel P_n \{R_1, \ldots, R_n; HF_1, \ldots, HF_n\}$$

 where A_i, R_i, HF_i are (first order or DC) formulas of $\mathcal{V}(P_i)$ $(i = 1, \ldots, n)$ separately. The validity can be defined similarly.
- Note that we can essentially put A and R as parts of history formula HF like the form $\lceil A \rceil^0 {}^\frown HF {}^\frown \lceil R \rceil^0$.

Example

In **PLC**, we can specify the system as:

$$\{s = s_0 \wedge u = u_0 \wedge Ctrl(u_0, s_0), A_2\}\ PLC$$
$$\{R_1, R_2; (l = T)^\frown \lceil |\ s - s_{targ}\ | \leq \epsilon \rceil, HF_2\}$$

where $Ctrl(u, s)$ may express a controllable property, and the other formulas are not elaborated here.

Outline

General Rules

Monotonicity

If $\{A_1, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$,
and $A_i' \Rightarrow A_i, R_i \Rightarrow R_i', HF_i \Rightarrow HF_i' (i = 1, 2)$,
then $\{A_1', A_2'\}P_1 \parallel P_2\{R_1', R_2'; HF_1', HF_2'\}$

Case Analysis

If $\{A_{1i}, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$ $(i = 1, 2)$,
then $\{A_{11} \vee A_{12}, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$

If $\{A_1, A_{2i}\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$ $(i = 1, 2)$,
then $\{A_1, A_{21} \vee A_{22}\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$

General Rules

Monotonicity

If $\{A_1, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$,
and $A'_i \Rightarrow A_i, R_i \Rightarrow R'_i, HF_i \Rightarrow HF'_i (i = 1, 2)$,
then $\{A'_1, A'_2\}P_1 \parallel P_2\{R'_1, R'_2; HF'_1, HF'_2\}$

Case Analysis

If $\{A_{1i}, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$ $(i = 1, 2)$,
then $\{A_{11} \vee A_{12}, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$

If $\{A_1, A_{2i}\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$ $(i = 1, 2)$,
then $\{A_1, A_{21} \vee A_{22}\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$

General Rules (Cont'd)

Parallel vs Sequential

If $\{A_1, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$
then $\{A_i\}P_i\{R_i; HF_i\}$ $(i = 1, 2)$

If $\{A_i\}P_i\{R_i; HF_i\}$ $(i = 1, 2)$,
and P_i $(i = 1, 2)$ do not contain communication,
then $\{A_1, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$

Skip

$$\{A\}\textbf{skip}\{A; l = 0\},$$

Assignment

$$\{A[e/x]\}x := e\{A, \lceil x = e \rceil^0\}$$

General Rules (Cont'd)

Parallel vs Sequential

If $\{A_1, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$
then $\{A_i\}P_i\{R_i; HF_i\}$ $(i = 1, 2)$

If $\{A_i\}P_i\{R_i; HF_i\}$ $(i = 1, 2)$,
and P_i $(i = 1, 2)$ do not contain communication,
then $\{A_1, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$

Skip

$$\{A\}\textbf{skip}\{A; l = 0\},$$

Assignment

$$\{A[e/x]\}x := e\{A, \lceil x = e\rceil^0\}$$

General Rules (Cont'd)

Parallel vs Sequential

If $\{A_1, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$
then $\{A_i\}P_i\{R_i; HF_i\}$ $(i = 1, 2)$

If $\{A_i\}P_i\{R_i; HF_i\}$ $(i = 1, 2)$,
and P_i $(i = 1, 2)$ do not contain communication,
then $\{A_1, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$

Skip

$$\{A\}\textbf{skip}\{A; l = 0\},$$

Assignment

$$\{A[e/x]\}x := e\{A, \lceil x = e\rceil^0\}$$

Communication

If $\{A_1, A_2\} P_1 \parallel P_2 \{R_1, R_2; HF_1, HF_2\}$,
$R_1 \Rightarrow G(e)$, $HF_1 \Rightarrow \ell = c_1$, and $HF_2 \Rightarrow \ell = c_2$
then $\{A_1, A_2\}(P_1; ch!e) \parallel (P_2; ch?x)$

$$\{R_1, G(x) \wedge \exists x.R_2; HF_1{}^\frown(\lceil R_1 \rceil \wedge const(\mathcal{V}(P_1)) \wedge \ell = c - c_1),$$
$$(HF_2{}^\frown(\lceil R_2 \rceil \wedge const(\mathcal{V}(P_2)) \wedge \ell = c - c_2))^{<\frown}\lceil x = e \rceil^0\}$$

where $c = \max\{c_1, c_2\}$.

- A rule for the general case of communication

$$(P_1; \|_{i \in I} ch_i* \rightarrow Q_{1i}) \parallel (P_2; \|_{j \in J} ch_j* \rightarrow Q_{2j}),$$

where $ch_i* = \overline{ch_j*}$ for some $i \in I, j \in J$, can be defined similarly.

Communication

If $\{A_1, A_2\}P_1 \parallel P_2\{R_1, R_2; HF_1, HF_2\}$,
$R_1 \Rightarrow G(e), HF_1 \Rightarrow \ell = c_1$, and $HF_2 \Rightarrow \ell = c_2$
then $\{A_1, A_2\}(P_1; ch!e) \parallel (P_2; ch?x)$

$$\{R_1, G(x) \wedge \exists x.R_2; HF_1{}^\frown(\lceil R_1\rceil \wedge const(\mathcal{V}(P_1)) \wedge \ell = c - c_1),$$
$$(HF_2{}^\frown(\lceil R_2\rceil \wedge const(\mathcal{V}(P_2)) \wedge \ell = c - c_2))^{<\frown}\lceil x = e\rceil^0\}$$

where $c = \max\{c_1, c_2\}$.

- A rule for the general case of communication

$$(P_1; \llbracket_{i\in I} ch_i* \to Q_{1i}) \parallel (P_2; \llbracket_{j\in J} ch_j* \to Q_{2j}),$$

where $ch_i* = \overline{ch_j*}$ for some $i \in I, j \in J$, can be defined similarly.

Communication (cont'd)

Example

If

$$\{A_1, A_2\}P_1 \parallel P_2$$
$$\{y = 3, x = 1; ([y = 0] \wedge (l = 3))^\frown[y = 3]^0, [x = 0] \wedge (l = 5)^\frown[x = e]^0\},$$

we want to deduce through this rule

$$\{A_1, A_2\}P_1; ch!y \parallel P_2; ch?x\{R_3, R_4; HF_3, HF_4\}.$$

Since $(y = 3) \Rightarrow (3 = 3)$, $([y = 0] \wedge (l = 3))^\frown[y = 3]^0 \Rightarrow \ell = 3$, and $([x = 0] \wedge \ell = 5)^\frown[x = 1]^0 \Rightarrow \ell = 5$, we can conclude that R_3 is $y = 3$, R_4 is $x = 3$, HF_3 is
$((([y = 0] \wedge (l = 3))^\frown[y = 3]^{0\frown}([y = 3] \wedge const(\mathcal{V}(P_1) \cup \{y\}) \wedge \ell = 2)$, and HF_4 is $(\ell = 5^\frown[x = 1]^0)^{<\frown}[x = 3]^0$, which is equivalent to $\ell = 5^\frown[x = 3]^0$ by the definition of $HF^<$.

Continuous

If $Init \Rightarrow Inv$,
then $\{Init \wedge A\}\langle F(\dot{s}, s) = 0 \& B\rangle\{A \wedge \mathbf{Cl}(Inv) \wedge \mathbf{Cl}(\neg B); \lceil Inv \wedge A \wedge B \rceil\}$

If $\{A\}\langle F(\dot{s}, s) = 0 \& B\rangle\{R; HF\}$
and $\{A \wedge t = 0\}\langle (F(\dot{s}, s) = 0, \dot{t} = 1) \& B\rangle\{t = t_0 \wedge Rg(t_0), HF'\}$,
then $\{A\}\langle F(\dot{s}, s) = 0 \& B\rangle\{R; HF \wedge Rg(\ell)\}$

Continuous (cont'd)

Example

In **MA**, it is easy to see that $v \leq v_{ebi}$ is an invariant of $\langle(\dot{s} = v, \dot{v} = a) \wedge v < v_{ebi}\rangle$. Thus, by the continuous rule

$$\{(v = v_0 \leq v_{ebi})\}\langle(\dot{s} = v, \dot{v} = a)\&v < v_{ebi}\rangle$$
$$\{(v \leq v_{ebi}) \wedge (v \geq v_{ebi}); \lceil(v \leq v_{ebi}) \wedge (v < v_{ebi})\rceil\}$$

In addition, assume $p \geq a \geq w$, then

$$((v_0 + wt) \leq v \leq (v_0 + pt)) \wedge (v \leq v_{ebi})$$

is an invariant of $\langle(\dot{s} = v, \dot{v} = a, \dot{t} = 1)\&v < v_{ebi}\rangle$. So,

$$\{(v = v_0 \leq v_{ebi}) \wedge (t = 0)\}\langle(\dot{s} = v, \dot{v} = a, \dot{t} = 1)\&v < v_{ebi}\rangle$$
$$\{(v = v_{ebi}) \wedge ((v_0 + wt) \leq v \leq (v_0 + pt)) \wedge \frac{v_{ebi} - v_0}{w} \geq t \geq \frac{v_{ebi} - v_0}{p};$$
$$\lceil(v < v_{ebi}) \wedge ((v_0 + wt) \leq v \leq (v_0 + pt))\rceil\}$$

Therefore, assuming $(p \geq a \geq w)$ we can have

$$\{(v = v_0 \leq v_{ebi})\}\langle(\dot{s} = v, \dot{v} = a)\&v < v_{ebi}\rangle$$
$$\{(v = v_{ebi}); \lceil(v < v_{ebi})\rceil \wedge (\frac{v_{ebi} - v_0}{w} \geq l \geq \frac{v_{ebi} - v_0}{p})\}$$

Communication Interruption

Rule1: If

1. $\{A, A_R\}\, \langle F(\dot{s}, s) = 0 \& B\rangle \,\|\, R \{R, R_R; HF, HF_R\}$,
2. for all $i \in I$, $\{A, A_R\}\, ch_i{*} \,\|\, R\{R_i, R_R^i; HF_i, HF_R^i\}$,
3. $HF \Rightarrow \ell = x$, $\wedge_{i\in I}(HF_i \Rightarrow \ell = x_i) \wedge x < x_i$,

then

$$\{A, A_R\}\, \langle F(\dot{s}, s) = 0 \& B\rangle \unrhd [\![_{i\in I}(ch_i{*} \to Q_i) \,\|\, R \\ \{R, R_R; HF, HF_R\}$$

Rule 2: Assume $j \in I$. If

1. $\{A, A_R\}\, \langle F(\dot{s}, s) = 0 \& B\rangle \,\|\, R_1; \overline{ch_j{*}} \to R_2 \{R, R_R; HF, HF_R\}$,
2. for all $i \in I$, $\{A, A_R\}\, ch_i{*} \,\|\, R_1; \overline{ch_j{*}}\, \{R_i, R_R^i; HF_i, HF_R^i\}$,
3. $HF \Rightarrow \ell = x$, $\wedge_{i\in I} HF_i \Rightarrow \ell = x_i$, and $x_j \le x \wedge \wedge_{i\neq j} x_j \le x_i$,
4. $HF \Rightarrow (\ell = x_j \wedge HF_s)^\frown \lceil G(s_0)\rceil^0$,
5. $\{R_j \wedge G(s_0), R_R^j\}\, Q_j \,\|\, R_2 \{R^f, R_R^f; HF^f, HF_R^f\}$,

then

$$\{A, A_R\}\, \langle F(\dot{s}, s) = 0 \& B\rangle \unrhd [\![_{i\in I}(ch_i{*} \to Q_i) \,\|\, R_1; \overline{ch_j{*}} \to R_2 \\ \{R^f, R_R^f; ((HF_s{}^\frown \lceil G(s_0)\rceil^0) \wedge HF_j)^\frown HF^f, HF_R^j{}^\frown HF_R^f\}$$

Communication Interruption

Rule1: If

1. $\{A, A_R\}\,\langle F(\dot{s}, s) = 0\&B\rangle \parallel R\{R, R_R; HF, HF_R\}$,
2. for all $i \in I$, $\{A, A_R\}\; ch_i* \parallel R\{R_i, R_R^i; HF_i, HF_R^i\}$,
3. $HF \Rightarrow \ell = x$, $\wedge_{i\in I}(HF_i \Rightarrow \ell = x_i) \wedge x < x_i$,

then

$$\{A, A_R\}\,\langle F(\dot{s}, s) = 0\&B\rangle \trianglerighteq \|_{i\in I}(ch_i* \to Q_i) \parallel R\;\{R, R_R; HF, HF_R\}$$

Rule 2: Assume $j \in I$. If

1. $\{A, A_R\}\,\langle F(\dot{s}, s) = 0\&B\rangle \parallel R_1; \overline{ch_j*} \to R_2\{R, R_R; HF, HF_R\}$,
2. for all $i \in I$, $\{A, A_R\}\; ch_i* \parallel R_1; \overline{ch_j*}\,\{R_i, R_R^i; HF_i, HF_R^i\}$,
3. $HF \Rightarrow \ell = x$, $\wedge_{i\in I} HF_i \Rightarrow \ell = x_i$, and $x_j \le x \wedge \wedge_{i\neq j} x_j \le x_i$,
4. $HF \Rightarrow (\ell = x_j \wedge HF_s)^\frown\lceil G(s_0)\rceil^0$,
5. $\{R_j \wedge G(s_0), R_R^j\}Q_j \parallel R_2\{R^f, R_R^f; HF^f, HF_R^f\}$,

then

$$\{A, A_R\}\,\langle F(\dot{s}, s) = 0\&B\rangle \trianglerighteq \|_{i\in I}(ch_i* \to Q_i) \parallel R_1; \overline{ch_j*} \to R_2\;\{R^f, R_R^f; ((HF_s{}^\frown\lceil G(s_0)\rceil^0) \wedge HF_j)^\frown HF^f, HF_R^j{}^\frown HF_R^f\}$$

Sequential, Internal Choice, Repetition

Sequential

If $\{A_1\}P_1\{R_1; HF_1\}$, and $\{R_1\}P_2\{R_2; HF_2\}$
then $\{A_1\}P_1; P_2\{R_2; HF_1^{<}{}^\frown HF_2\}$.

Internal Choice

If $\{A\}P_1\{R_1; HF_1\}$ and $\{A\}P_2\{R_2; HF_2\}$,
then $\{A\}P_1 \sqcup P_2\{R_1 \vee R_2; HF_1 \vee HF_2\}$.

Repetition

$$\begin{array}{l}
\text{If } \{A_1, A_2\}P_1 \parallel P_2\{A_1, A_2; HF_1, HF_2\}, \\
\quad HF_i \Rightarrow (D_i \wedge (l = T))\ (i = 1, 2,\ T \geq 0), \\
\text{and } D_i^\frown D_i \Rightarrow D_i, \\
\text{then } \{A_1, A_2\}P_1^* \parallel P_2^*\{A_1, A_2; \ell = 0 \vee D_1, \ell = 0 \vee D_2\}
\end{array}$$

where T is the time consumed by both P_1 and P_2 that can guarantee the synchronisation of the starting point of each repetition.

Sequential, Internal Choice, Repetition

Sequential

If $\{A_1\}P_1\{R_1; HF_1\}$, and $\{R_1\}P_2\{R_2; HF_2\}$
then $\{A_1\}P_1; P_2\{R_2; HF_1^{<}{}^\frown HF_2\}$.

Internal Choice

If $\{A\}P_1\{R_1; HF_1\}$ and $\{A\}P_2\{R_2; HF_2\}$,
then $\{A\}P_1 \sqcup P_2\{R_1 \vee R_2; HF_1 \vee HF_2\}$.

Repetition

If $\{A_1, A_2\}P_1 \parallel P_2\{A_1, A_2; HF_1, HF_2\}$,
$HF_i \Rightarrow (D_i \wedge (l = T))\ (i = 1, 2,\ T \geq 0)$,
and $D_i{}^\frown D_i \Rightarrow D_i$,
then $\{A_1, A_2\}P_1^* \parallel P_2^*\{A_1, A_2; \ell = 0 \vee D_1, \ell = 0 \vee D_2\}$

where T is the time consumed by both P_1 and P_2 that can guarantee the synchronisation of the starting point of each repetition.

Sequential, Internal Choice, Repetition

Sequential

If $\{A_1\}P_1\{R_1; HF_1\}$, and $\{R_1\}P_2\{R_2; HF_2\}$
then $\{A_1\}P_1; P_2\{R_2; HF_1^{<\frown}HF_2\}$.

Internal Choice

If $\{A\}P_1\{R_1; HF_1\}$ and $\{A\}P_2\{R_2; HF_2\}$,
then $\{A\}P_1 \sqcup P_2\{R_1 \vee R_2; HF_1 \vee HF_2\}$.

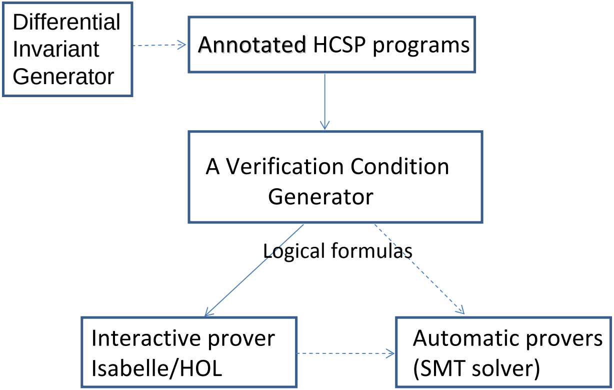
Repetition

$$
\begin{array}{l}
\text{If } \{A_1, A_2\}P_1 \parallel P_2\{A_1, A_2; HF_1, HF_2\}, \\
\quad HF_i \;\Rightarrow\; (D_i \wedge (l = T))\ (i = 1, 2,\ T \geq 0), \\
\text{and } D_i^{\frown}D_i \;\Rightarrow\; D_i, \\
\text{then } \{A_1, A_2\}P_1^* \parallel P_2^*\{A_1, A_2; \ell = 0 \vee D_1, \ell = 0 \vee D_2\}
\end{array}
$$

where T is the time consumed by both P_1 and P_2 that can guarantee the synchronisation of the starting point of each repetition.

Outline

Verification Architecture of HHL Prover

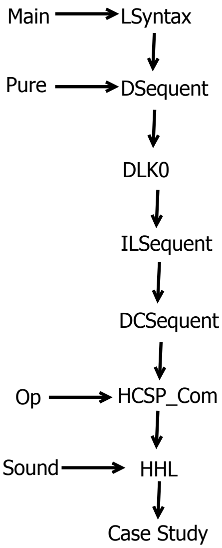


Theorem Proving

The implementation of a theorem prover for verifying annotated HCSP processes in a proof assistant can be divided into the following steps:

- Encode HCSP language (syntax and semantics);
- Encode assertion languages (syntax, semantics and proof system);
- Encode Hybrid Hoare Logic (syntax, inference rules and validity), and based on this, design a verification generator.
  - The VCG reduces an annotated HCSP process to a logical formula written by the assertion languages, whose validity is equivalent to the validity of the original annotated process.
- Discharge the validity of the logical formulas by interactive or automatic theorem proving.
  - Integration with SMT solvers (for solving formulas).

Sketch of Our Implementation in Deep Encoding



- Main, Pure: Isabelle theories, for all the basic predefined theories like arithmetic, lists, sets, etc; and the meta-logic HOL.
- LSyntax: Syntax for expressions and FOL.
- DSequent, DLK0: Sequent calculus based proof system for FOL.
- ILSequent, DCSequent: Syntax and proof systems for IL and DC.
- HCSP_Com, Op: Syntax and semantics for HCSP language.
- HHL, Sound: Proof system for hybrid Hoare Logic, and soundness.
- Case Study: Case studies, always including HHL theory as a parent.

Syntax for Expressions and FOL

Expression

datatype exp = RVar string | SVar string | BVar string | Real real
| String string | Bool bool | exp $+$ exp | exp $-$ exp | exp $*$ exp

FOL

datatype fform = True | False | exp $=$ exp | exp $<$ exp
| $\neg$ fform | fform $\vee$ fform | $\forall$ string fform

The other constructs of FOL can be derived from the above ones.

Syntax for Temporal Expressions and DC

Temporal Expression

datatype dexp $= \ell$ | Real real

DC

datatype dform = True | False | dexp = dexp | dexp < dexp
| $\neg$ dform | dform $\vee$ dform | $\forall$ string dform
| pf fform | dform $\frown$ dform

Derived Operators of DC

consts high :: fform $\Rightarrow$ dform
high S == $\neg$ (True $\frown$ pf ($\neg$ S) $\frown$ ℓ > Real 0)

Sequent Calculus Proof System of Assertion Languages

- A sequent is a pair written as $\Gamma \vdash \Delta$, where Γ and Δ are sequences of formulas. Usually, P, Q are used to represent a logical formula, $\$H$, $\$E$ arbitrary sequences of logical formulas.
- The axiom and proof system of FOL and DC can be defined by a set of sequent rules, each of which is a relation between a (possibly empty) sequence of sequents and a single sequent
  - For example, the right introducing rule for conjunction in FOL:

$$\frac{\Gamma \vdash P, \Delta \quad \Gamma \vdash Q, \Delta}{\Gamma \vdash P \wedge Q, \Delta}$$

For sequent calculus based proof system,

- Backward proof search is applied.
- Be widely used in mechanized deductive reasoning (via pattern matching of goals).

Sequent Calculus Proof System of FOL

- Isabelle Library includes the Sequent Calculus Proof System for Classical FOL with Equation, in theory *LK*. Our encoding of FOL proof system is built from it directly.
- We need to add extra lemmas for reasoning about explicit arithmetic formulas.
- To reuse the solvers for bool, the built-in type of Isabelle logical formulas, we define an equivalent relation between the validity of formulas of `fform` and of *bool*:

$$formT(f :: fform) \Leftrightarrow \vdash f$$

where formT :: fform $\Rightarrow$ bool

Sequent Calculus Proof System of DC

- First, define the deductive system for the first-order constructs of *dform*, which can be taken directly from the one built for *fform*;
- Second, define the deductive system related to the new temporal modalities for DC, including ℓ, *pf*, and $^\frown$.

For the second step, we transform the proof system in Hilbert style defined in [Zhou& Hansen 2004] to sequent calculus style.

- Not a direct translation.
- Related work [Heilmann99, Rasmussen02].

For instance,

LI : \$H, P $\vdash$ \$E $\Rightarrow$ \$H, P$^\frown$(ℓ = Real 0) $\vdash$ \$E
RI : \$H $\vdash$ P, \$E $\Rightarrow$ \$H $\vdash$ P$^\frown$(ℓ = Real 0), \$E
encodes the axiom: $P \rightarrow P^\frown(\ell = 0)$.

HCSP Language

We define HCSP constructs as a datatype *proc*. Each construct of HCSP is encoded correspondingly, except for the two special cases:

- The differential equation $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle$ is encoded as `<Inv&B> : Rg`, where `Inv` represents the differential invariant, `B` the domain constraint, and `Rg` the range of execution time, of the continuous respectively.
- The sequential composition $P; Q$ is encoded as `P; mid; Q`, where `P` and `Q` represent the encodings of P and Q respectively, and `mid` is added to represent the intermediate assertions between P and Q.
  - `mid` is added for reducing proof of sequential composition to the ones of its components; and we can remove `mid` and instead deduce it directly based on HHL proof system.

Semantics

Encoding of states, configurations, and transition rules, that are defined in previous sections

Hybrid Hoare Logic

Specification

The specification for process `P` is represented as `{Pre} P {Post;HF}`, where `Pre` and `Post` are implemented as formulas of type `fform`, and `HF` of type `dform`, `P` as process of type `proc`, and it corresponds to a truth proposition.

Inference Rules

All the inference rules are encoded as theorems. A verification condition generator (VCG) can be formed by structural composition of these theorems (to be shown in detail).

Validity

All the inference rules are sound with respect to the semantics of HCSP, thus the correctness of the VCG is guaranteed.

Verification Condition Generator

A common approach for proving $\{p\}P\{q;\ HF\}$:

- to calculate weakest precondition (Dijkstra's) backwards from q, denoted by $WP(P, q)$, or to calculate strongest postcondition forwards from p, denoted by $SP(P, p)$;
- to calculate strongest history formula forwards from p, denoted by $SH(P, p)$.

Then, $\{p\}P\{q;\ HF\}$ iff one of the following conditions holds:

- $p \Rightarrow WP(P, q)$ and $SH(P, p) \Rightarrow HF$
- $SP(P, p) \Rightarrow q$ and $SH(P, p) \Rightarrow HF$

Our proof system of HCSP provides rules for designing the verification condition generator for HCSP, by combining the two methods.

Verification Condition Generator : Assignment

According to HHL proof system,

$$\{A[e/x]\}x := e\{A;\ \lceil x = e \rceil^0\}$$

The weakest precondition of $x := e$ with respect to postcondition A is $A[e/x]$, and the strongest history formula is $\lceil x = e \rceil^0$.

Thus, to prove $\{p\}x := e\{q;\ HF\}$, we can prove

$$p \Rightarrow q[e/x] \wedge (\lceil x = e \rceil^0 \Rightarrow HF)$$

instead. The validity of them is equivalent.

Verification Condition Generator : Sequential Composition

According to HHL proof system, to prove

$$\{p\}P; (m, H); Q\{q; H^\frown G\}$$

we can prove `{p} P {m;H}` and `{m} Q {q;G}` instead.

Notice that the intermediate assertions are annotated (i.e., `(m, H)` above) to refer to the postcondition and the history formula of the first component.

As an alternative approach,

- deduce `m` as weakest precondition $WP(Q, q)$ or strongest postcondition $SP(P, p)$, and `H` as strongest history formula $SH(P, p)$.

Verification Condition Generator : Continuous

According to HHL proof system,

$$\{Init \wedge A\} \langle \mathcal{F}(\dot{s}, s) = 0 \& B : Inv \rangle \{A \wedge Cl(Inv) \wedge Cl(\neg B); \\ (l = 0) \vee \lceil Inv \wedge A \wedge B \rceil\}$$

To prove $\{p_1 \wedge p_2\}\langle \mathcal{F}(\dot{s}, s) = 0 \& B : Inv \rangle \{q;\ HF\}$, we can prove

- $p_2 \Rightarrow Inv$
- $p_1 \wedge close(Inv) \wedge close(\neg B) \Rightarrow q$
- $(l = 0) \vee \lceil Inv \wedge Pre \wedge B \rceil \Rightarrow HF$

instead.

- *Inv* is assumed and annotated.
- We are considering to integrate the VCG with the differential invariant generator. The above formulas will then be the constraints for calculating the differential invariant.

Verification Condition Generator

The cases for other constructs, including communication, communication interrupt, repetition, etc, are also implemented.

The soundness of the verification condition generator is proved in Isabelle/HOL.

Finally, an annotated HCSP process is transformed into logical formulas (including FOL and DC formulas), with equivalent validity.

- Interactive theorem proving, by applying axioms corresponding to proof systems of FOL and DC manually.
- Integration with SMT solvers, especially for deciding FOL formulas.

Outline

Chinese Train Control System Level 3

- **The Chinese Train Control System (CTCS)** at level 3 (CTCS-3) is an informal specification of Chinese high speed train that ensures safety and high throughput of trains.
- For historical reasons, CTCS currently contains two levels: level 2 and level 3.
- 14 scenarios
  - **Movement authority**
  - **Level transition** (upgrade, degrade)
  - **Mode transition** (**FS** to **CO**)
  - ...

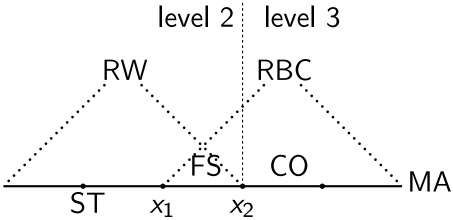
Chinese Train Control System Level 3

- **The Chinese Train Control System (CTCS)** at level 3 (CTCS-3) is an informal specification of Chinese high speed train that ensures safety and high throughput of trains.
- For historical reasons, CTCS currently contains two levels: level 2 and level 3.
- 14 scenarios
  - **Movement authority**
  - **Level transition** (upgrade, degrade)
  - **Mode transition** (**FS** to **CO**)
  - · · ·

Chinese Train Control System Level 3

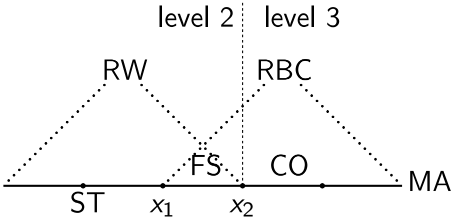
- **The Chinese Train Control System (CTCS)** at level 3 (CTCS-3) is an informal specification of Chinese high speed train that ensures safety and high throughput of trains.
- For historical reasons, CTCS currently contains two levels: level 2 and level 3.
- 14 scenarios
  - **Movement authority**
  - **Level transition** (upgrade, degrade)
  - **Mode transition** (**FS** to **CO**)
  - ...

Case Study: A Combined Scenario of CTC-3



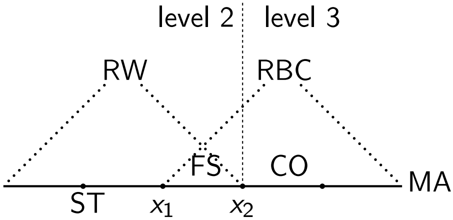
- FS to CO transition at x_2
- CTCS-2 to CTCS-3 transition at x_2
- Modelling + annotated property, and then verification by theorem proving.

Case Study: A Combined Scenario of CTC-3



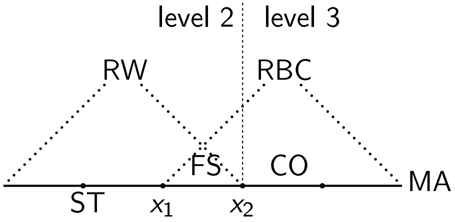
- **FS** to **CO** transition at x_2
- CTCS-2 to CTCS-3 transition at x_2
- Modelling + annotated property, and then verification by theorem proving.

Case Study: A Combined Scenario of CTC-3



- **FS** to **CO** transition at x_2
- **CTCS-2** to **CTCS-3** transition at x_2
- Modelling + annotated property, and then verification by theorem proving.

Case Study: A Combined Scenario of CTC-3



- **FS** to **CO** transition at x_2
- **CTCS-2** to **CTCS-3** transition at x_2
- Modelling + annotated property, and then verification by theorem proving.

Movement Authority Scenario

- The train applies for **MA** from **RBC** in **CTCS-3** or **TCC** in **CTCS-2**. It is only permitted to move within the **MA** it owns.
- Each **MA** is modelled as a sequence of tuples of form $\langle (s, v1, v2, mode), \cdots, (s, v1, v2, mode) \rangle$
- Static profile and dynamic profile

Movement Authority Scenario

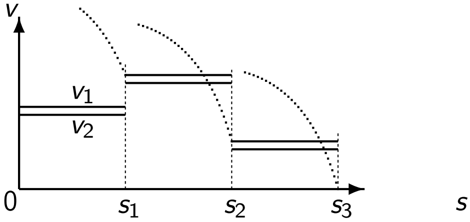
- The train applies for **MA** from **RBC** in **CTCS-3** or **TCC** in **CTCS-2**. It is only permitted to move within the **MA** it owns.
- Each **MA** is modelled as a sequence of tuples of form $\langle (s, v1, v2, mode), \cdots, (s, v1, v2, mode) \rangle$
- Static profile and dynamic profile

Movement Authority Scenario

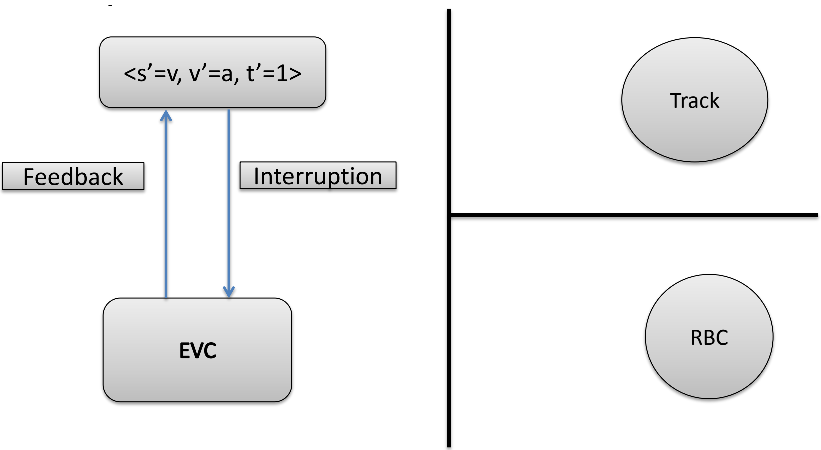
- The train applies for **MA** from **RBC** in **CTCS-3** or **TCC** in **CTCS-2**. It is only permitted to move within the **MA** it owns.
- Each **MA** is modelled as a sequence of tuples of form $\langle (s, v1, v2, mode), \cdots, (s, v1, v2, mode) \rangle$
- Static profile and dynamic profile

Movement Authority Scenario

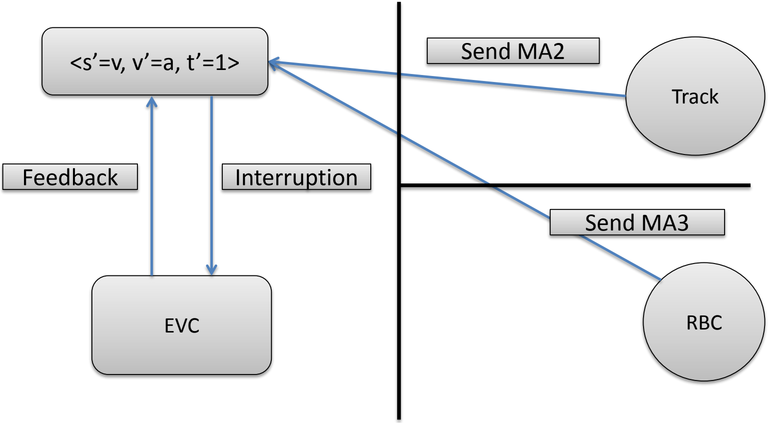
- The train applies for **MA** from **RBC** in **CTCS-3** or **TCC** in **CTCS-2**. It is only permitted to move within the **MA** it owns.
- Each **MA** is modelled as a sequence of tuples of form $\langle(s, v1, v2, mode), \cdots, (s, v1, v2, mode)\rangle$
- Static profile and dynamic profile



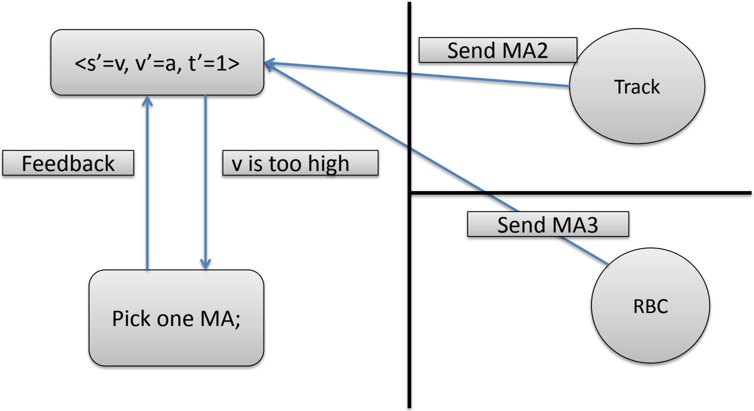
Movement Authority Scenario



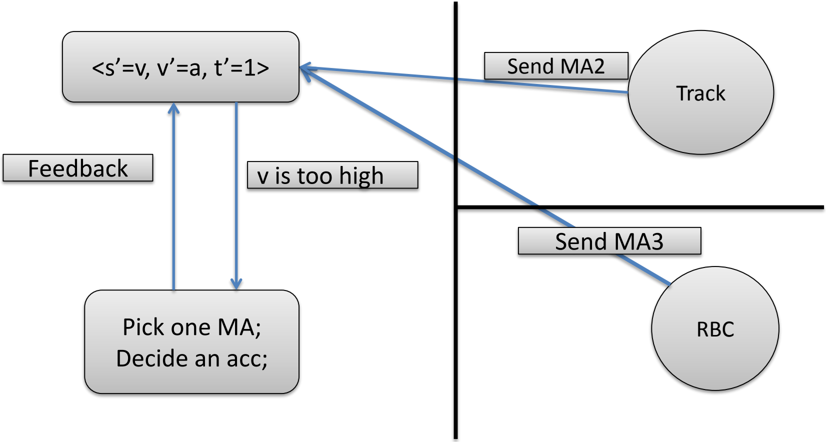
Movement Authority Scenario



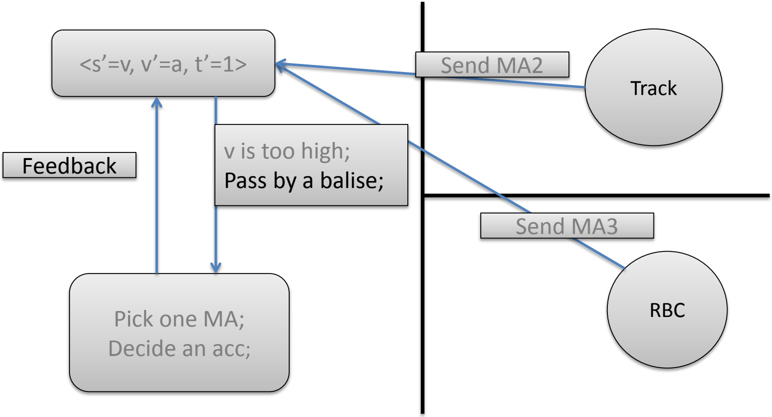
Movement Authority Scenario



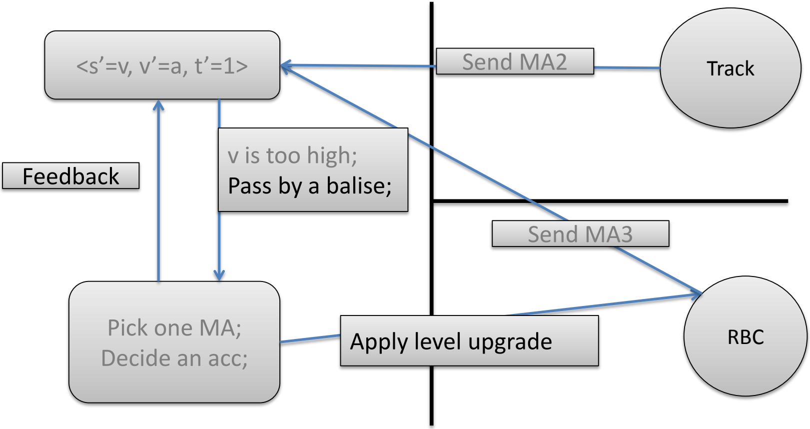
Movement Authority Scenario



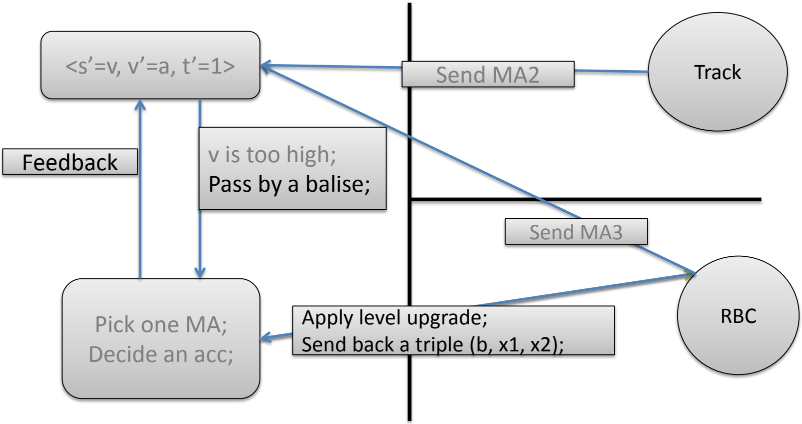
Level Upgrade Scenario



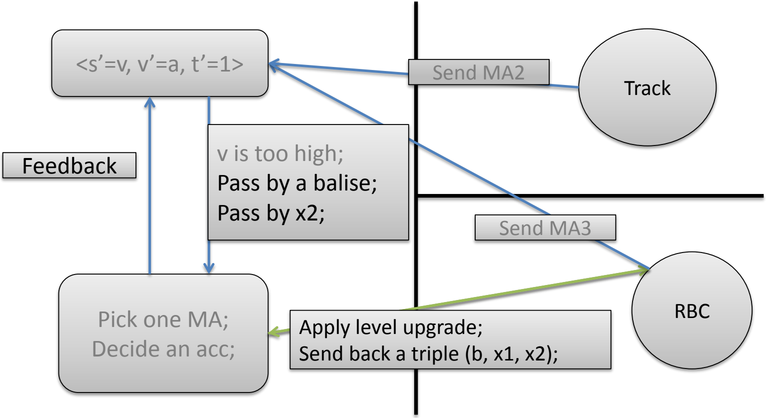
Level Upgrade Scenario



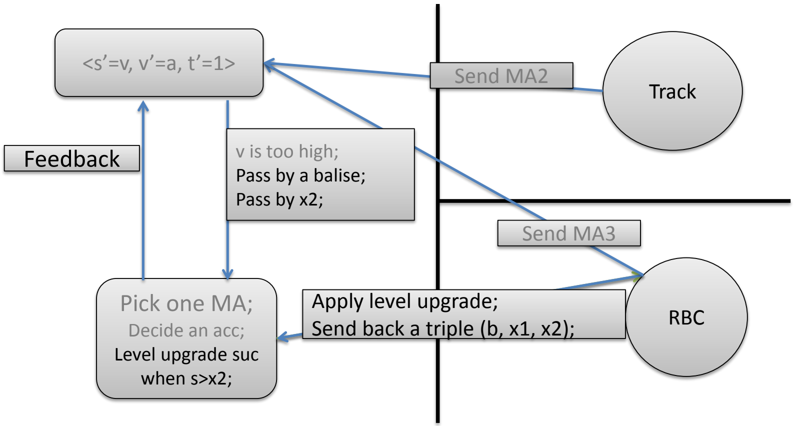
Level Upgrade Scenario



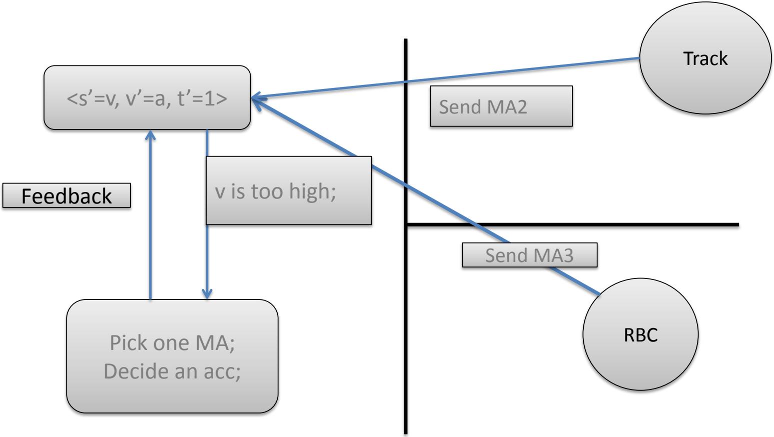
Level Upgrade Scenario



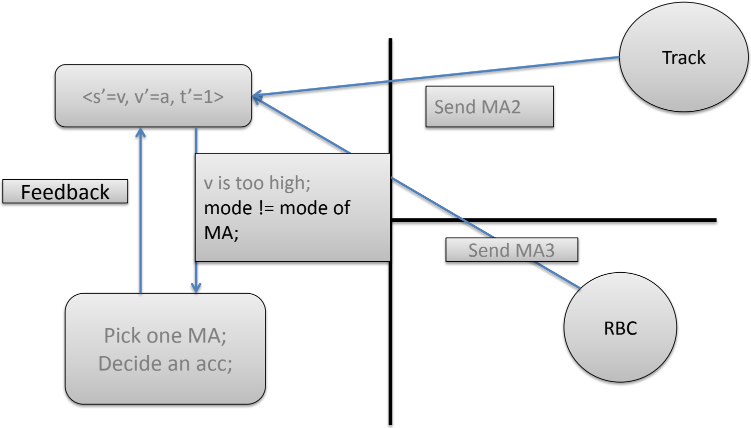
Level Upgrade Scenario



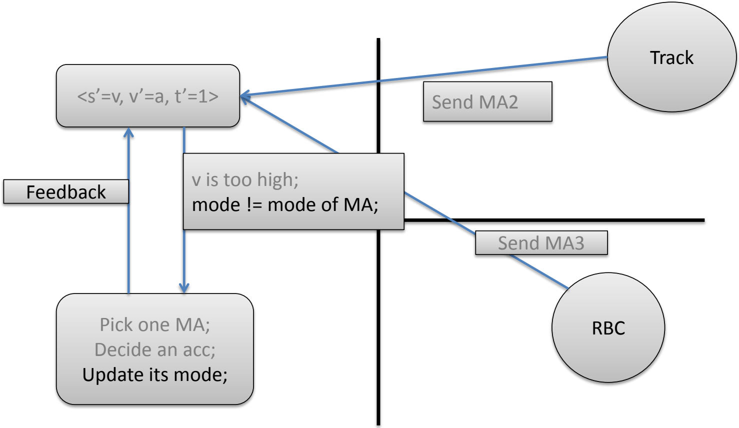
Mode Conversion on Level 2: FS to CO



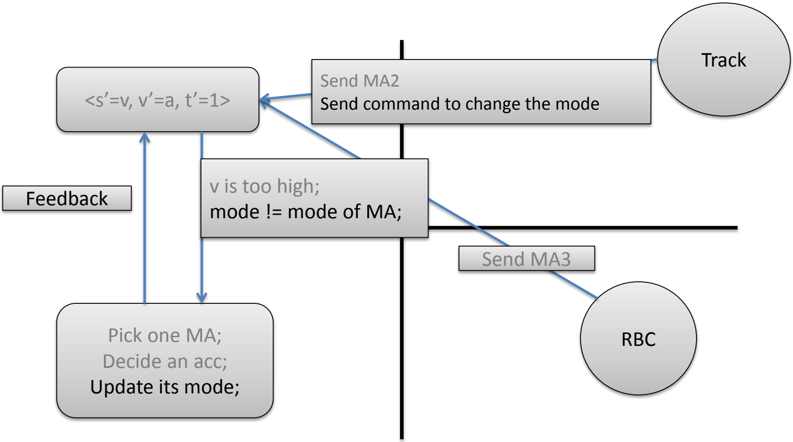
Mode Conversion on Level 2: FS to CO



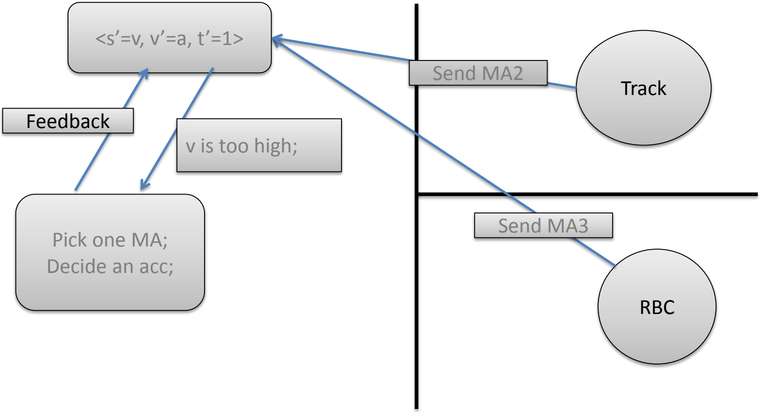
Mode Conversion on Level 2: FS to CO



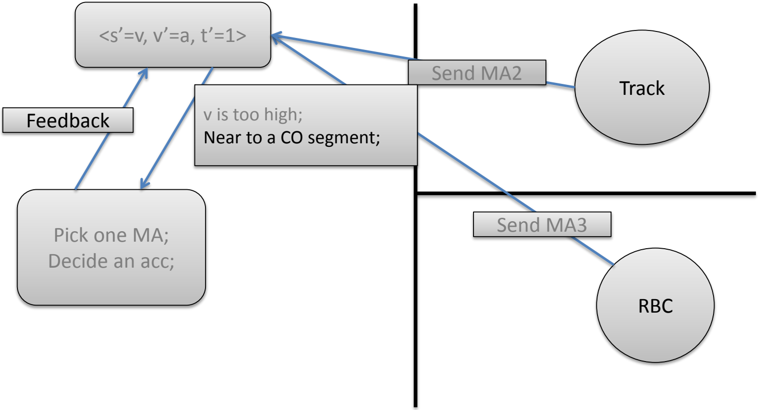
Mode Conversion on Level 2: FS to CO



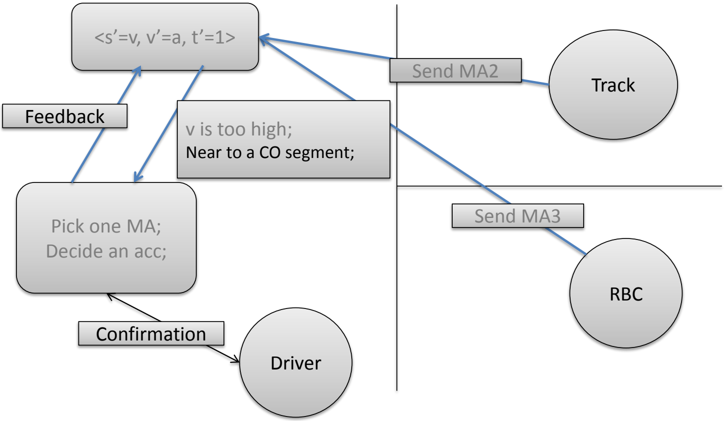
Mode Conversion on Level 3: FS to CO



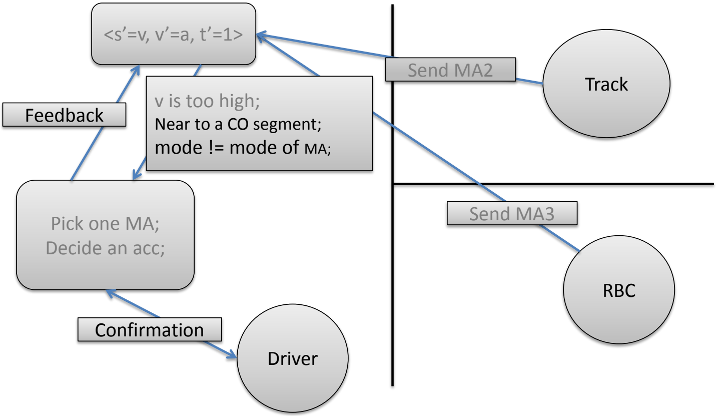
Mode Conversion on Level 3: FS to CO



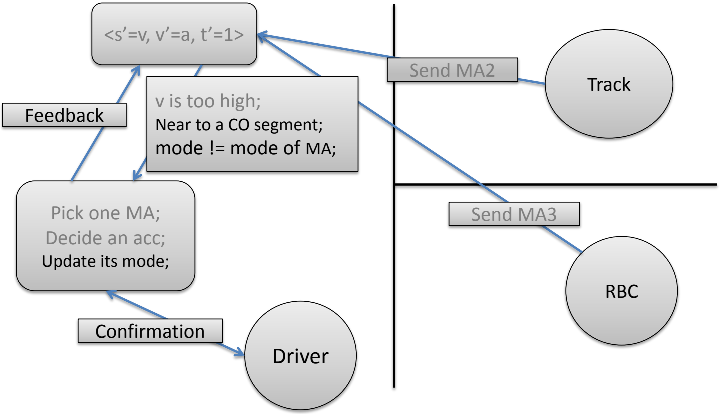
Mode Conversion on Level 3: FS to CO



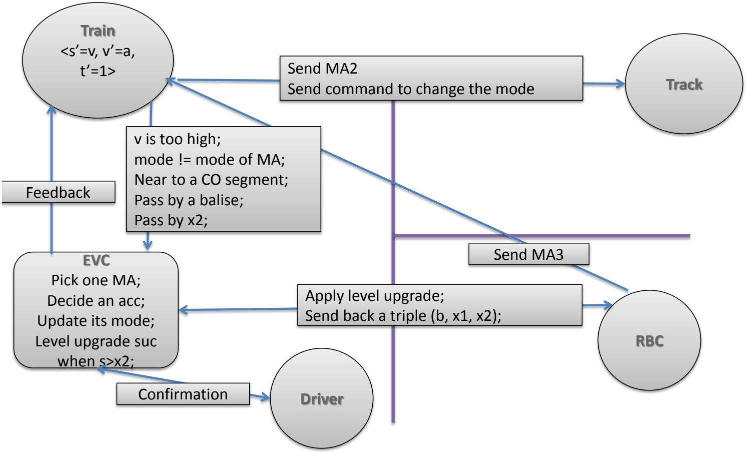
Mode Conversion on Level 3: FS to CO



Mode Conversion on Level 3: FS to CO



Combined Scenario



Problem of the Combined Scenario

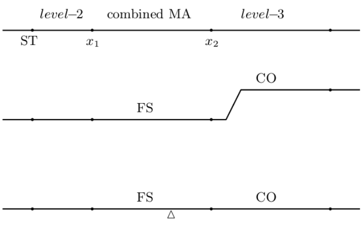
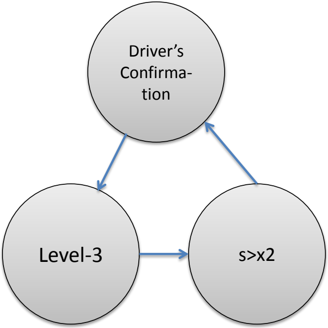


Fig. 1. The scenarios.



Formal Model of the Scenario

The overall train control system:

$$\begin{aligned}
System &\mathrel{\widehat{=}} Train^* \parallel Driver^*_{mc} \parallel RBC^*_{lu} \parallel RBC^*_{ma} \parallel TCC^* \\
Train &\mathrel{\widehat{=}} \langle \dot{s} = v, \dot{v} = a, \dot{t} = 1 \& \; B_0 \wedge B_1 \wedge B_2 \wedge B_3 \wedge B_4 \wedge B_5 \wedge B_6 \wedge B_7 \rangle; P_{train} \\
P_{train} &\mathrel{\widehat{=}} Q1_{comp}; Q2_{comp}; Q3_{comp}; Q4_{comp}; Q5_{comp}
\end{aligned}$$

Movement authority scenario:

$$\begin{aligned}
B_0 &\mathrel{\widehat{=}} (v \geq 0 \vee a \geq 0 \vee t < Temp + T_{delay}) \\
B_1 &\mathrel{\widehat{=}} (\forall seg : MA \,.\, v < seg.v_2) \vee a < 0 \vee t < Temp' + T_{delay} \\
B_2 &\mathrel{\widehat{=}} (\forall seg : MA \,.\, v < seg.v_1 \wedge v^2 + 2b\, s < next(seg).v_1^2 + 2b\, seg.e) \\
&\quad \vee a = -b \\
B_7 &\mathrel{\widehat{=}} (s <= hd(MA).e) \\
Q1_{comp} &\mathrel{\widehat{=}} \neg B_0 \rightarrow (Temp := t; \sqcup_{\{0<=c<=A\}} a := c); \\
&\quad \neg B_1 \rightarrow (Temp' := t; \sqcup_{\{-b<=c<0\}} a := c); \\
&\quad \neg B_2 \rightarrow a := -b; CH_{b2}!\neg B_7; CH_{b3}!\neg B_7; \\
&\quad \neg B_7 \rightarrow (CH_{eoa2}!getEoA(rMA2); ch_{ma2}?rMA2; \\
&\qquad\qquad CH_{eoa3}!getEoA(rMA3); ch_{ma3}?rMA3; \\
&\qquad\qquad MA := comb(rMA2, rMA3))
\end{aligned}$$

Formal Model of the Scenario (Cont'd)

***Level transition scenario*:**

$$
\begin{array}{ll}
B_3 & \widehat{=} level \neq 2 \ \vee \ s \neq n * \delta \\
B_4 & \widehat{=} level \neq 2.5 \ \vee \ s \leq LU.x_2 \\
Q2_{comp} & \widehat{=} \neg B_3 \rightarrow (CH_{LUA}!; CH_{LU}?LU; LU.b \rightarrow level = 2.5; n = n + 1); \\
Q3_{comp} & \widehat{=} \neg B_4 \rightarrow level := 3 \\
RBC_{lu} & \widehat{=} CH_{LUA}?; \sqcup_{b_{LU} \in \{true, false\}} CH_{LU}!(b, x_1, x_2)
\end{array}
$$

***Mode transition scenario*:**

$$
\begin{array}{ll}
B_5 & \widehat{=} mode = hd(MA).mode \\
Q4_{comp} & \widehat{=} \neg B_5 \rightarrow mode := hd(MA).mode \\
B_6 & \widehat{=} level \neq 3 \vee CO \neq hd(tl(MA)).mode \vee hd(MA).e - s > 300 \\
 & \vee t < Temp + T_{delay} \\
Q5_{comp} & \widehat{=} CH_{win}!\neg B_6; \neg B_6 \rightarrow Temp := t; CH_{DC}?b_{rConf}; b_{rConf} \rightarrow coma(MA) \\
Driver_{mc} & \widehat{=} CH_{win}?b_{win}; b_{win} \rightarrow \sqcup_{b_{sConf} \in \{true, false\}} CH_{DC}!b_{sConf} \\
TCC & \widehat{=} CH_{b2}?b2; b2 \rightarrow (CH_{eoa2}?eoa2; ch_{ma2}!setMA2(eoa2)) \\
RBC_{ma} & \widehat{=} CH_{b3}?b3; b3 \rightarrow (CH_{eoa3}?eoa3; ch_{ma3}!setMA3(eoa3))
\end{array}
$$

Specification and Verification of the Scenario

The specification for train to be verified:

$$\{Pre\}\,Train\{Post;\ HF\}$$

where

$$
\begin{aligned}
Pre &\stackrel{\text{def}}{=} (x_2 - s > Real\ 300) \wedge (level = Real\ 2.5) \\
&\quad \wedge (fst(snd(snd(hd(MA)))) = x_2) \\
&\quad \wedge (snd(snd(snd(hd(MA)))) = String\ \text{“}FS'') \\
&\quad \wedge (snd(snd(snd(hd(tl(MA))))) = String\ \text{“}CO'') \\
&\quad \wedge (fst(hd(MA)) = Real\ 0) \wedge (fst(snd(hd(MA))) = Real\ 0) \\
Post &\stackrel{\text{def}}{=} s <= x_2 \\
HF &\stackrel{\text{def}}{=} (\ell = Real\ 0) \vee (high(Pre \wedge s <= x_2))
\end{aligned}
$$

The specification has been proved in HHL prover. This shows that the train will never exceed x_2.

Remarks

- An implementation of HHL in Isabelle/HOL
  - Can be found at https://github.com/iscas/HHL_prover
  - In progress
- Application to modelling and verification of Chinese Train Control Systems
  - Formal model with HCSP
  - Justification of formal model: from Simulink graphical model to HCSP model [Zou et al 2013b]; from HCSP model to Simulink model (on-going)
  - Simulation by Simulink
  - Verification by HHL Prover
  - The bug reported here is first found by the testing group of Beijing Jiaotong University, analyzed and proved by us [Zou et al 2013a]
  - More than 6 bugs are found in CTCS-3 in other similar combined scenarios first by us
- Application to modelling and verification of Spacecraft Control Systems

→ Demo

Remarks

- An implementation of HHL in Isabelle/HOL
  - Can be found at https://github.com/iscas/HHL_prover
  - In progress
- Application to modelling and verification of Chinese Train Control Systems
  - Formal model with HCSP
  - Justification of formal model: from Simulink graphical model to HCSP model [Zou et al 2013b]; from HCSP model to Simulink model (on-going)
  - Simulation by Simulink
  - Verification by HHL Prover
  - The bug reported here is first found by the testing group of Beijing Jiaotong University, analyzed and proved by us [Zou et al 2013a]
  - More than 6 bugs are found in CTCS-3 in other similar combined scenarios first by us
- Application to modelling and verification of Spacecraft Control Systems

→ **Demo**

Remarks

- An implementation of HHL in Isabelle/HOL
  - Can be found at https://github.com/iscas/HHL_prover
  - In progress
- Application to modelling and verification of Chinese Train Control Systems
  - Formal model with HCSP
  - Justification of formal model: from Simulink graphical model to HCSP model [Zou et al 2013b]; from HCSP model to Simulink model (on-going)
  - Simulation by Simulink
  - Verification by HHL Prover
  - The bug reported here is first found by the testing group of Beijing Jiaotong University, analyzed and proved by us [Zou et al 2013a]
  - More than 6 bugs are found in CTCS-3 in other similar combined scenarios first by us
- Application to modelling and verification of Spacecraft Control Systems

→ Demo

Remarks

- An implementation of HHL in Isabelle/HOL
  - Can be found at https://github.com/iscas/HHL_prover
  - In progress
- Application to modelling and verification of Chinese Train Control Systems
  - Formal model with HCSP
  - Justification of formal model: from Simulink graphical model to HCSP model [Zou et al 2013b]; from HCSP model to Simulink model (on-going)
  - Simulation by Simulink
  - Verification by HHL Prover
  - The bug reported here is first found by the testing group of Beijing Jiaotong University, analyzed and proved by us [Zou et al 2013a]
  - More than 6 bugs are found in CTCS-3 in other similar combined scenarios first by us
- Application to modelling and verification of Spacecraft Control Systems

→ **Demo**