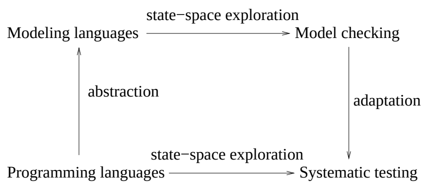
Lecture 4:

May/Must Abstraction-Based Software Model Checking

Patrice Godefroid

Microsoft Research



Automatic Abstraction

Current automatic abstraction tools typically proceed as follows:

- Given a concrete program C, they generate an abstract program A such that "A simulates C".
- For any $\forall$-properties ϕ, $A \models \phi$ implies $C \models \phi$.

Limitations:

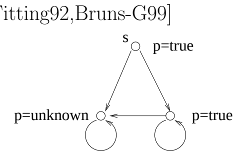
- Restricted to $\forall$-properties (no existential properties).
- $A \not\models \phi$ does not imply anything about C!
- Could the analysis be more precise for a comparable cost?

A Solution: use 3-Valued Models [Bruns-G99]

Use richer models A that distinguish what is *true*, *false* and unknown ($\perp$) of C.

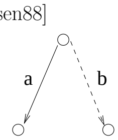
Example: partial Kripke structure (PKS) [Fitting92,Bruns-G99]

- A Kripke structure where propositions can be *true*, *false* or $\perp$.



Example: Modal Transition System [Larsen-Thomsen88]

- A LTS with $\xrightarrow{may}$ and $\xrightarrow{must}$ transitions such that $\xrightarrow{must} \subseteq \xrightarrow{may}$.



Example: Kripke Modal Transition System [Huth-Jagadeesan-Schmidt01]

- A PKS with $\xrightarrow{may}$ and $\xrightarrow{must}$ transitions such that $\xrightarrow{must} \subseteq \xrightarrow{may}$.

These models are all equally expressive [G-Jagadeesan03].

Other examples: extended transition systems [Milner81],...

3-Valued Temporal Logics

Reasoning about 3-valued models requires 3-valued TL.

Example: 3-valued Propositional Modal Logic $\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid AX\phi$

Semantics: (extension of Kleene's strong 3-valued PL)

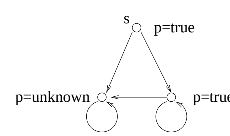
$[(M, s) \models p] = L(s, p)$

$[(M, s) \models \neg\phi] = \text{comp}([(M, s) \models \phi])$
where comp maps *true* $\mapsto$ *false*, *false* $\mapsto$ *true*, and $\perp \mapsto \perp$

$[(M, s) \models \phi_1 \wedge \phi_2] = min([(M, s) \models \phi_1], [(M, s) \models \phi_2])$
with min defined with *false* $<\perp<$ *true* ("truth" ordering)

$$[(M,s) \models AX\phi] = \begin{cases} \mathit{true} & \text{if } \forall s' : s \xrightarrow{may} s' \Rightarrow [(M,s') \models \phi] = \mathit{true} \\ \mathit{false} & \text{if } \exists s' : s \xrightarrow{must} s' \wedge [(M,s') \models \phi] = \mathit{false} \\ \perp & \text{otherwise} \end{cases}$$

- Ex: $[(M, s) \models p] = \mathit{true}$
- Ex: $[(M, s) \models AXp] = \perp$



Completeness Preorder

To measure the *completeness* of models (aka, *refinement* preorder, or *abstraction*$^{-1}$.)

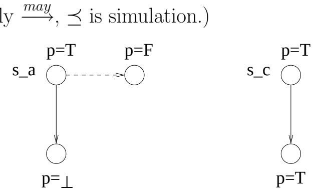
Let $\leq$ be the "information" ordering on truth values in which $\perp \leq \mathit{true}$ and $\perp \leq \mathit{false}$.

Definition: The *completeness preorder* $\preceq$ is the greatest relation $\preceq \subseteq S \times S$ such that $s_a \preceq s_c$ implies the following:

- $\forall p \in P : L_A(s_a, p) \leq L_C(s_c, p)$,
- if $s_a \xrightarrow{must}_A s'_a$, there is some $s'_c \in S_C$ such that $s_c \xrightarrow{must}_C s'_c$ and $s'_a \preceq s'_c$,
- if $s_c \xrightarrow{may}_C s'_c$, there is some $s'_a \in S_A$ such that $s_a \xrightarrow{may}_A s'_a$ and $s'_a \preceq s'_c$.

(Note: if no $\perp$ and only $\xrightarrow{may}$, $\preceq$ is simulation.)

Example:



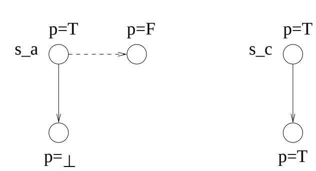
Logical Characterization of Completeness Preorder

Theorem: Let Φ denote the set of all formulas of 3-valued propositional modal logic. Then

$$s_a \preceq s_c \text{ iff } (\forall \phi \in \Phi : [s_a \models \phi] \leq [s_c \models \phi]).$$

Thus, models that are "more complete" with respect to $\preceq$ have more definite properties with respect to $\leq$.

Example:



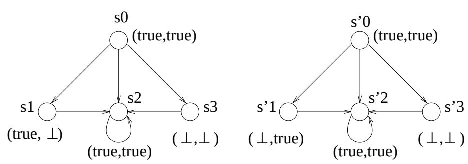
Completeness Preorder (Continued)

Corollary:
Let Φ denote the set of all formulas of 3-valued propositional modal logic. Then

$$(\forall \phi \in \Phi : [(M_1, s_1) \models \phi] = [(M_2, s_2) \models \phi]) \text{ iff}$$
$$(s_1 \preceq s_2 \text{ and } s_2 \preceq s_1).$$

Note: If s_1 and s_2 are bisimilar, this implies $s_1 \preceq s_2$ and $s_2 \preceq s_1$,
but $s_1 \preceq s_2$ and $s_2 \preceq s_1$ does not imply s_1 and $s2$ are bisimilar! [Bruns-G99]

Example: s_0 and s_0' are not bisimilar, but cannot be distinguished by any formula of 3-valued propositional modal logic.



3-Valued Model Checking

Problem: Given a state s of a 3-valued model M and a formula ϕ,
how to compute the value $[(M, s) \models \phi]$?

Theorem: [Bruns-G00] The model-checking problem for a 3-valued temporal logic can be reduced to two model-checking problems for the corresponding 2-valued logic.

STEP 1: complete M into two "extreme" complete Kripke structures, called the **optimistic** and **pessimistic** completions:

- Extend P to P' such that, for every $p \in P$ there exists a $\bar{p} \in P'$ such that $L(s, p) = \text{comp}(L(s, \bar{p}))$ for all s in S.
- $M_o = (S, L_o, \xrightarrow{must})$ with

$$L_o(s,p) \stackrel{\text{def}}{=} \begin{cases} true & \text{if } L(s,p) = \perp \\ L(s,p) & \text{otherwise} \end{cases}$$

- $M_p = (S, L_p, \xrightarrow{may})$ with

$$L_p(s,p) \stackrel{\text{def}}{=} \begin{cases} false & \text{if } L(s,p) = \perp \\ L(s,p) & \text{otherwise} \end{cases}$$

3-Valued Model Checking (Continued)

STEP 2: transform ϕ to its positive form $T(\phi)$ with $T(\neg p) = \bar{p}$.

STEP 3: evaluate $T(\phi)$ on M_o and M_p using traditional 2-valued model checking, and combine the results:

$$[(M, s) \models \phi] = \begin{cases} true & \text{if } (M_p, s) \models T(\phi) \\ false & \text{if } (M_o, s) \not\models T(\phi) \\ \perp & \text{otherwise} \end{cases}$$

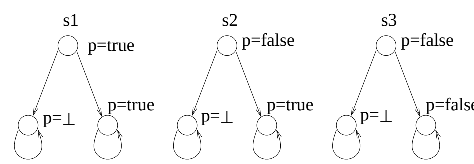
This can be done using existing model-checking tools!

Corollary: 3-valued model checking has the same complexity as traditional 2-valued model checking.

Examples

Application:
Generation of a partial Kripke structure from a partial state-space exploration such that, by construction, $s_0' \preceq s_0$ [Bruns-G99].

Examples:



- $[s_1 \models A(true\,\mathcal{U}\,p)] = true$
- $[s_2 \models A(true\,\mathcal{U}\,p)] = \perp$
- $[s_3 \models A(true\,\mathcal{U}\,p)] = false$

New 3-Valued Semantics

Observation: One can argue that the previous semantics returns $\perp$ more often than it should...

Example: In a state s_a where $p = \perp$ and $q = true$,

$$[s_a \models \ q \wedge (p \vee \neg p)\ ] = \perp$$

while the same formula is *true* in every complete state s_c such that $s_a \preceq s_c$!

New 3-valued "thorough" semantics: [Bruns-G00]

$$[(M, s) \models \phi]_t = \begin{cases} true & \text{if } (M', s') \models \phi \text{ for all } (M', s') : s \preceq s' \\ false & \text{if } (M', s') \not\models \phi \text{ for all } (M', s') : s \preceq s' \\ \perp & \text{otherwise} \end{cases}$$

Is model checking more expensive with this semantics?

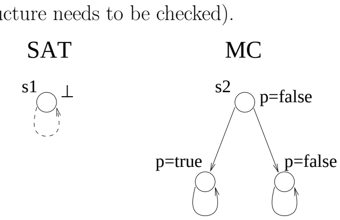
YES! Indeed, in general, one needs to solve two

Generalized Model-Checking Problems

Generalized Model Checking [Bruns-G00]

Definition: Given a state s of a model M and a formula ϕ of a temporal logic L, is there a state s' of a complete system M' such that $s \preceq s'$ and $(M', s') \models \phi$?

This **generalized model-checking problem** is thus a generalization of both **satisfiability** (all Kripke structures are potential solutions) and **model checking** (a single Kripke structure needs to be checked).



Theorem: The satisfiability problem for a temporal logic L is reducible (in linear-time and logarithmic space) to the generalized model-checking problem for L.

Thus, GMC is as hard as satisfiability. Is it harder?

Branching-Time Temporal Logics

Theorem: (CTL) Given a state s_0 of partial Kripke structure $M = (S, L, \mathcal{R})$ and a CTL formula ϕ, one can construct an alternating Büchi word automaton $A_{(M,s_0),\phi}$ over a 1-letter alphabet with at most $O(|S| \cdot 2^{O(|\phi|)})$ states such that

$$(\exists (M', s_0') : s_0 \preceq s_0' \text{ and } (M', s_0') \models \phi) \text{ iff } \mathcal{L}(A_{(M,s_0),\phi}) \neq \emptyset.$$

Corollary: if such a M' exists, there exists one with at most $|S| \cdot 2^{O(|\phi|)}$ states.

Theorem: The generalized model-checking problem for a state s_0 of a partial Kripke structure $M = (S, L, \mathcal{R})$ and a CTL formula ϕ can be decided in time $O(|S|^2 \cdot 2^{O(|\phi|)})$.

Theorem: The generalized model-checking problem for CTL is EXPTIME-complete.

Theorem: (Summary) Let L denote propositional logic, propositional modal logic, CTL, or any branching-time logic including CTL (such as CTL* or the modal μ-calculus). The generalized model-checking problem for the logic L has the same complexity as the satisfiability problem for L. [Bruns-G00]

Linear-Time Temporal Logics [G-Piterman09]

Theorem: (LTL) Given a state s_0 of partial Kripke structure $M = (S, L, \mathcal{R})$ and an LTL formula ϕ, one can construct an alternating parity word automaton $A_{(M,s_0),\phi}$ over a 1-letter alphabet with at most $O(|S| \cdot 2^{2^{|\phi| log(|\phi|)}})$ states and $2^{O(|\phi|)}$ priorities such that

$$(\exists (M', s_0') : s_0 \preceq s_0' \text{ and } (M', s_0') \models \phi) \text{ iff } \mathcal{L}(A_{(M,s_0),\phi}) \neq \emptyset.$$

Theorem: The generalized model-checking problem for a state s_0 of a partial Kripke structure $M = (S, L, R)$ and an LTL formula ϕ can be decided in time polynomial in $|S|$ and doubly exponential in $|\phi|$.

Theorem: The generalized model-checking problem for linear-time temporal logic is 2EXPTIME-complete.

For LTL, generalized model checking is thus **harder** than satisfiability and model checking! (both of these problems are PSPACE-complete for LTL)

Note: similar phenomenon for "realizability" and "synthesis" for LTL specifications [Abadi-Lamport-Wolper89, Pnueli-Rosner89].

Summary on Complexity in $|\phi|$

Model Checking: (3-valued semantics)

- MC can be reduced to two 2-valued MC problems.
- MC has the same complexity as 2-valued MC.

Generalized Model Checking: (thorough 3-val. sem.)

- For BTL, GMC has the same complexity as satisfiablity.
- For LTL, GMC is harder than satisfiablity and MC.

Logic	MC	SAT	GMC
PL	Linear	NP-Complete	NP-Complete
PML	Linear	PSPACE-Complete	PSPACE-Complete
CTL	Linear	EXPTIME-Complete	EXPTIME-Complete
μ-calculus	NP∩co-NP	EXPTIME-Complete	EXPTIME-Complete
LTL	PSPACE-Complete	PSPACE-Complete	2EXPTIME-Complete

Complexity of GMC in $|M|$

For CTL, GMC can be solved in time quadratic in $|M|$ [Bruns-G00].

For LTL, GMC can be solved in time polynomial in $|M|$ [G-Piterman00]:

- *linear* for safety ($\Box p$) and weak (recognizable by DWW) properties
- *quadratic* for response ($\Box(p \rightarrow \Diamond q)$), persistence ($\Diamond \Box p$) and generalized reactivity[1] properties [Kesten-Piterman-Pnueli03]

Note: for CTL and LTL, GMC is PTIME-hard in $|M|$ while MC is NLOGSPACE-complete in $|M|$ [G03]

Application: Automatic Abstraction

Idea: Given a concrete system C, if $C \models \phi$ cannot be decided, generate a (smaller) abstraction A and check $A \models \phi$ instead.

Example: predicate abstraction

- Let $\psi_1, \ldots, \psi_n$ be n predicates on variables of C.
- Abstract states are vectors of n bits b_i.
- A concrete state c is abstracted by an abstract state

$$[c] = (b_1, \ldots, b_n) \text{ iff } \forall 1 \leq i \leq n : b_i = \psi_i(c).$$

State of the art: A is a traditional 2-valued model with

$$(c_1 \rightarrow c_2) \Rightarrow ([c_1] \rightarrow [c_2]).$$

In other words, A **simulates** C. Remember, this implies:

- If ϕ is a $\forall$-property, $A \models \phi$ implies $C \models \phi$,
- but $A \not\models \phi$ does not imply anything about C!

Automatic Abstraction Revisited

Observation: A should really be a 3-valued model!

For instance, A can be represented by a modal transition system.

Abstraction relation:

1. $(c_1 \rightarrow c_2) \Rightarrow ([c_1] \rightarrow_{\text{may}} [c_2])$
2. $(\forall c_i \in [a] : \exists c_i \rightarrow c_j \wedge c_j \in [a']) \Rightarrow ([a] \rightarrow_{\text{must}} [a'])$

By construction, $A \preceq C$.

Computing an MTS A using (1)+(2) can be done at the same computational cost (same complexity) as computing a "conservative" abstraction (simulation) using (1) alone: (2) can be built by dualizing all the steps necessary to build (1).

This is shown for predicate and cartesian abstraction in [G-Huth-Jagadeesan01].

Automatic Abstraction Process

Traditional iterative abstraction procedure:

1. Abstract: compute M_A that simulates M_C.
2. Check: given a universal property ϕ, check $M_A \models \phi$.
   - if $M_A \models \phi$: stop (the property is proved: $M_C \models \phi$).
   - if $M_A \not\models \phi$: go to Step 3.
3. Refine: refine M_A. Then go to Step 1.

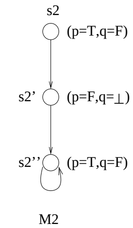
New procedure for automatic abstraction: (3 improvements)

1. Abstract: compute M_A such that $M_A \preceq M_C$ (same cost as above [GHJ01])
2. Check: given *any* property ϕ,
   1. (3-valued model checking) compute $[M_A \models \phi]$.
      - if $[M_A \models \phi] = $ *true* or *false*: stop .
      - if $[M_A \models \phi] = \perp$, continue.
   2. (generalized model checking) compute $[M_A \models \phi]_t$.
      - if $[M_A \models \phi]_t = $ *true* or *false*: stop .
      - if $[M_A \models \phi]_t = \perp$, go to Step 3.
3. Refine: refine M_A. Then go to Step 1.

Example

Predicate abstraction with p : "is x odd?" and q : "is y odd?" such that $M_2 \preceq C_2$:

```
program C2() {
  x,y = 1,0;
  x,y = 2*f(x),f(y);
  x,y = 1,0;
}
```



For $\phi_2 = \Diamond q \wedge \Box(p \vee \neg q)$, $[(M_2, s_2) \models \phi_2] = \perp$, but $[(M_2, s_2) \models \phi_2]_t = \mathit{false}$ (i.e., there does not exist a concretization of (M_2, s_2) that satisfies ϕ_2).

Thus, GMC is more precise than MC in this case.

(Same for the safety property $\phi'_2 = Xq \wedge \Box(p \vee \neg q)$.)

Precision of GMC Vs. MC

How often is GMC more precise than MC? See [G-Huth05]:

- Studies when it is possible to reduce GMC(M, ϕ) to MC(M, ϕ').
- ϕ' is called a *semantic minimization* of ϕ.
- Shows that PL (already known), PML, and μ-calculus are closed under semantic minimization, but not LTL, CTL or CTL*.
- Identifies *self-minimizing* formulas, i.e., ϕ's for which GMC$(M, \phi) =$ MC(M, ϕ)
  - semantically (using automata-theoretic techniques, EXPTIME-hard in $|\phi|$ for μ-calculus) and
  - syntactically (sufficient criterion only, linear in $|\phi|$).
- Ex (syntactic): Any formula that does not contain any atomic proposition in mixed polarity (in its negation normal form) is self-minimizing.
- Note: the converse is not true (e.g., $(\neg q_1 \vee q_2) \wedge (\neg q_2 \vee q_1)$ is self-minimizing).
- For any self-minimizing formula, GMC and MC have the same precision.
- Good news: many frequent formulas of practical interest are self-minimizing, and MC is as precise as GMC for those.

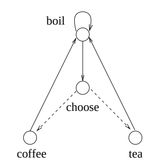
3-Valued Abstractions for Open Systems

Open system: system interacting with its environment.

Module Checking (ModC) [Kupferman-Vardi96]: given an open system M and a formula ϕ, does M satisfy ϕ in *all possible environments*?

Example: (vending machine)
is it always possible for M to eventually serve tea?

- MC(M, AGEF tea) $= \mathit{true}$
- ModC(M, AGEF tea) $= \mathit{false}$!



Generalized Module Checking (GModC) [G03]: given A and ϕ, does there exist a concretization C of A such that C satisfies ϕ in all possible environments?

Two simulataneous games here: one with the environment, one with $\perp$ values...

Yet, GModC can be solved at the same cost as GMC (for LTL and BTL) [G03].

3-Valued Abstractions for Games

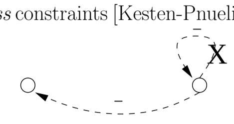
Study abstractions of games where moves of each player can now be abstracted, while preserving winning strategies of *both* players [de Alfaro-G-Jagadeesan04]:

- An abstraction of a game is now a game where each player has both may and must moves (yielding may/must strategies).
- Completeness preorder is now an *alternating refinement* relation, logically characterized by *3-valued alternating μ-calculus* [Alur-Henzinger-Kupferman02].

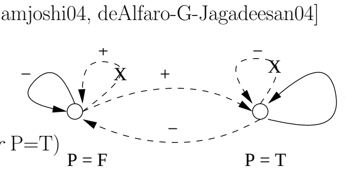
Semantic Completeness

Given any infinite-state system C and property ϕ, if C satisfies ϕ, then there exists a finite-state abstraction A such that A satisfies ϕ.

- **LTL:** True if abstractions extended with *fairness* constraints [Kesten-Pnueli00]
  Example:
  - var x; actions (-) if (x$\geq$0) x:=x-1;
  - property: AF($\neg$ P) with P= (x$\geq$ 0)
  - self-loop is unfair (models termination)



- **μ-calculus:** True if must transitions can be *nondeterministic* [Larsen-Xinxin90] (aka *hyper-must*) [Namjoshi03, Dams-Namjoshi04, deAlfaro-G-Jagadeesan04]
  Example: [Namjoshi03]
  - var x;
    actions (-) x:=x-1; (+) x:=x+1;
  - property: EF(P) with P= (x$\geq$ 0)
  - hyper-must transition (P=F, P=F *or* P=T)



  The construction of abstraction is now *compositional* (cf. [G-Huth-Jagadeesan01], [Shoham-Grumberg04], [de Alfaro-G-Jagadeesan04]).

Conclusions

3-Valued models and logics can be used to check *any property*, while *guaranteeing soundness of counter-examples*.

Generalized Model Checking means checking whether there exists a concretization of an abstraction that satisfies a temporal logic formula.

It can be used to improve precision of automatic abstraction, for a reasonable cost:

- Cost can be higher in the size of the formula...
  but only worst-case and formulas are short.
- Cost can be higher (e.g., quadratic) in the size of the model...
  but is the same (linear) for popular properties (e.g., safety).

In an "abstract-check-refine" procedure, GMC is only polynomial in the size of the abstraction, and may prevent the unnecessary generation and analysis of possibly exponentially larger refinements of that abstraction.

In practice, use first a syntactic formula check for self-minimization:
MC has then the same precision as GMC (often the case).

Some Other Related Work

"Mixed transition systems" [Dams-Gerth-Grumberg94]

- Intuitively, a mixed transition system is an MTS without the constraint $\xrightarrow{must} \subseteq \xrightarrow{may}$.
- More expressive: some mixed TS cannot be refined into any complete system.
- Still, their goal is very similar (i.e., design may/must abstractions for MC).

"Extended transition systems" [Milner81] = LTS + "divergence predicate"

- In [Bruns-G99], it is shown that 3-valued Hennessy-Milner Logic logically characterizes the "divergence preorder" [Milner81,Walker90].
- Close correspondence with Plotkin's intuitionistic modal logic (inspired Bruns-G00 reduction from 3-val to 2-val MC).

3-Valued logic for program analysis: [Sagiv-Reps-Wilhelm99] shape graphs, first-order 3-valued logic, "focussing",... (roughly inspired the beginning of this work but technical details are fairly different – e.g., no 3-valued abstraction on control)

Conservative abstraction for the full mu-calculus: [Saidi-Shankar99]

Yasm: 3-valued predicate abstraction tool [Gurfinkel-Chechik]

Abstraction refinement for 3-valued models [Shoham-Grumberg] etc.