Bisimulation and Logic
Lecture 4

Colin Stirling

Laboratory for Foundations of Computer Science (LFCS)
School of Informatics
Edinburgh University

Summer School on Model Checking
Ziyu Hotel, Beijing
Oct 11–16 2010

Bisimulation equivalence

- A binary relation B between processes is a bisimulation provided that, whenever $(E, F) \in B$ and $a \in \mathsf{A}$,

Bisimulation equivalence

- A binary relation B between processes is a bisimulation provided that, whenever $(E, F) \in B$ and $a \in \mathsf{A}$,
- if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' such that $(E', F') \in B$ and

Bisimulation equivalence

- A binary relation B between processes is a bisimulation provided that, whenever $(E, F) \in B$ and $a \in \mathrm{A}$,
- if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' such that $(E', F') \in B$ and
- if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some E' such that $(E', F') \in B$

Bisimulation equivalence

- A binary relation B between processes is a bisimulation provided that, whenever $(E, F) \in B$ and $a \in \mathsf{A}$,
- if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' such that $(E', F') \in B$ and
- if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some E' such that $(E', F') \in B$
- E and F are bisimulation equivalent (or bisimilar) if there is a bisimulation relation B such that $(E, F) \in B$.

Bisimulation equivalence

- A binary relation B between processes is a bisimulation provided that, whenever $(E, F) \in B$ and $a \in \mathsf{A}$,
- if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' such that $(E', F') \in B$ and
- if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some E' such that $(E', F') \in B$
- E and F are bisimulation equivalent (or bisimilar) if there is a bisimulation relation B such that $(E, F) \in B$.
- We write $E \sim F$ if E and F are bisimilar

Temporal operators as fixed points

Here $-$ represents any action

- $\mathsf{E}(\Phi\,\mathsf{U}\,\Psi) \equiv \Psi \;\vee\; (\Phi \wedge \langle - \rangle \mathrm{E}(\Phi\,\mathrm{U}\,\Psi))$
- $\mathsf{A}(\Phi\,\mathsf{U}\,\Psi) \equiv \Psi \;\vee\; (\Phi \wedge \langle - \rangle \mathtt{tt} \wedge [-]\mathrm{A}(\Phi\,\mathrm{U}\,\Psi))$

Syntactically: property X such that

1. $X \equiv \Psi \vee (\Phi \wedge \langle - \rangle X)$
2. $X \equiv \Psi \vee (\Phi \wedge \langle - \rangle \mathtt{tt} \wedge [-]X)$

Temporal Operators as Fixed points

Semantically: set of states or processes $S = f(S)$ where f is

- $\lambda x. \| \Psi \vee (\Phi \wedge \langle - \rangle x) \|$
- $\lambda x. \| \Psi \vee (\Phi \wedge \langle - \rangle \mathtt{tt} \wedge [-] x) \|$

If $S = f(S)$ then S is a fixed point of f.

In both cases f is monotonic: $S \subseteq S' \rightarrow f(S) \subseteq f(S')$

f is essentially modal (using $\langle - \rangle$ and $[-]$)

Bisimilarity as a fixed point

$\sim$ is a binary relation on processes, $\sim \subseteq S \times S$
Semantically a fixed point solution of equation:

$$R = f(R)$$

where $R \subseteq S \times S$ and f is the (monotonic) function

$$\lambda R'.\lambda xy.\forall a \in A$$

$$\begin{array}{l} \text{if } x \xrightarrow{a} x' \text{ then } \exists y'.y \xrightarrow{a} y' \text{ and } x' R' y' \text{ and} \\ \text{if } y \xrightarrow{a} y' \text{ then } \exists x'.x \xrightarrow{a} x' \text{ and } x' R' y' \end{array}$$

Summary: fixed points

S is a prefixed point of f, if $f(S) \subseteq S$

S is a postfixed point of f, if $S \subseteq f(S)$

Proposition If f is monotonic (w.r.t $\subseteq$) then f

- has a least fixed point, $\bigcap\{S : f(S) \subseteq S\}$
- has a greatest fixed point, $\bigcup\{S : S \subseteq f(S)\}$

Fixed points

Assume g is monotonic

$$\begin{array}{lccl} \text{least fixed point} & \mu g & = & \bigcap\{S : g(S) \subseteq S\} \\ \text{greatest fixed point} & \nu g & = & \bigcup\{S : S \subseteq g(S)\} \end{array}$$

Bisimilarity is a greatest fixed point

$$\sim = \bigcup\{R : R \text{ is a bisimulation}\}$$

Approximants I

Let $\nu^i g$ for $i \geq 0$ be defined as follows where S' is the full starting set $\nu^0 g = S'$ and $\nu^{i+1} g = g(\nu^i g)$.

- $\nu^{i+1} g \subseteq \nu^i g$ for all i
- Moreover, $\nu g \subseteq \nu^i g$ for all i

$$\begin{array}{ccccccccc} \nu^0 g & \supseteq & \nu^1 g & \supseteq & \dots & \supseteq & \nu^i g & \supseteq & \dots \\ \cup & & \cup & & & & \cup & & \\ \nu g & & \nu g & & \dots & & \nu g & & \dots \end{array}$$

- If $\nu^i g = \nu^{i+1} g$, then νg is $\nu^i g$

Approximants II

- If S' is not a finite set, then use ordinals
$$0, 1, \ldots, \omega, \omega + 1, \ldots, \omega + \omega, \omega + \omega + 1, \ldots$$
- ω is the initial limit ordinal
- $\nu^0 g = S'$ and $\nu^{\alpha+1} g = g(\nu^\alpha g)$ and if λ is a limit ordinal

$$\nu^\lambda g \quad = \quad \bigcap \{\nu^\alpha g \,:\, \alpha < \lambda\}$$

Approximants III

$$\begin{array}{ccccccccccc}
\nu^0 g & \supseteq & \dots & \supseteq & \nu^\omega g & \supseteq & \nu^{\omega+1} g & \supseteq & \dots \\
\cup & & & & \cup & & \cup & & \\
\nu g & & \dots & & \nu g & & \nu g & & \dots
\end{array}$$

The fixed point νg appears somewhere in the sequence, at the first point when $\nu^\alpha g = \nu^{\alpha+1} g$

Approximants IV

- $\mu^0 g = \emptyset$ and $\mu^{\alpha+1} g = g(\mu^\alpha g)$ and
  $\mu^\lambda g \quad = \quad \bigcup\{\mu^\alpha g \, : \, \alpha < \lambda\}$
- There is the following possibly increasing sequence of sets.

$$
\begin{array}{ccccccccc}
\mu g & & \ldots & & \mu g & & \mu g & & \ldots \\
\cup & & & & \cup & & \cup & & \\
\mu^0 g & \subseteq & \ldots & \subseteq & \mu^\omega g & \subseteq & \mu^{\omega+1} g & \subseteq & \ldots
\end{array}
$$

- The first time $\mu^\alpha g = \mu^{\alpha+1} g$ is μg

The bisimilarity problem

- Given: two processes E and F

The bisimilarity problem

- Given: two processes E and F
- Decide: is $E \sim F$? i.e., are E and F (strongly) bisimilar ?

The bisimilarity problem

- Given: two processes E and F
- Decide: is $E \sim F$? i.e., are E and F (strongly) bisimilar ?
- Assume both E and F are finite state

The bisimilarity problem

- Given: two processes E and F
- Decide: is $E \sim F$? i.e., are E and F (strongly) bisimilar ?
- Assume both E and F are finite state
- Restrict relations to subsets of $S \times S$, where S is processes in transition systems for E and F

The bisimilarity problem

- Given: two processes E and F
- Decide: is $E \sim F$? i.e., are E and F (strongly) bisimilar ?
- Assume both E and F are finite state
- Restrict relations to subsets of $S \times S$, where S is processes in transition systems for E and F
- Outline of the algorithm:

The bisimilarity problem

- Given: two processes E and F
- Decide: is $E \sim F$? i.e., are E and F (strongly) bisimilar ?
- Assume both E and F are finite state
- Restrict relations to subsets of $S \times S$, where S is processes in transition systems for E and F
- Outline of the algorithm:
  - Compute $\sim \subseteq S \times S$.

The bisimilarity problem

- Given: two processes E and F
- Decide: is $E \sim F$? i.e., are E and F (strongly) bisimilar ?
- Assume both E and F are finite state
- Restrict relations to subsets of $S \times S$, where S is processes in transition systems for E and F
- Outline of the algorithm:
  - Compute $\sim \subseteq S \times S$.
  - Check if $(E, F) \in \sim$.

Bisimilarity up to n

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.

Bisimilarity up to n

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:

Bisimilarity up to n

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:
- $E \sim_0 F$ for all E and F.

Bisimilarity up to n

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:
- $E \sim_0 F$ for all E and F.
- $E \sim_{n+1} F$ if and only if for every action a,

Bisimilarity up to n

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:
- $E \sim_0 F$ for all E and F.
- $E \sim_{n+1} F$ if and only if for every action a,
  - if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' such that $E' \sim_n F'$, and

Bisimilarity up to n

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:
- $E \sim_0 F$ for all E and F.
- $E \sim_{n+1} F$ if and only if for every action a,
  - if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' such that $E' \sim_n F'$, and
  - if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some E' such that $E' \sim_n F'$.

Bisimilarity up to n

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:
- $E \sim_0 F$ for all E and F.
- $E \sim_{n+1} F$ if and only if for every action a,
  - if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' such that $E' \sim_n F'$, and
  - if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some E' such that $E' \sim_n F'$.

$$
\begin{array}{ccc}
E & \sim_{n+1} & F \\
\downarrow a & & \downarrow a \\
E' & \sim_n & F'
\end{array}
$$

Key result

Proposition For all $n \geq 0$,

1. $\sim_n \supseteq \sim$,
2. $\sim_n \supseteq \sim_{n+1}$, and
3. If $\sim_n = \sim_{n+1}$, then $\sim_n = \sim$.

Scheme for the computation of $\sim$

- Compute $\sim_0, \sim_1, \sim_2, \ldots$ until $\sim_i = \sim_{i+1}$.

Scheme for the computation of $\sim$

- Compute $\sim_0, \sim_1, \sim_2, \ldots$ until $\sim_i = \sim_{i+1}$.
- Output $\sim_i$.

Scheme for the computation of $\sim$

- Compute $\sim_0, \sim_1, \sim_2, \ldots$ until $\sim_i = \sim_{i+1}$.
- Output $\sim_i$.
- Correctness: Part (3) of the Proposition.

Scheme for the computation of $\sim$

- Compute $\sim_0, \sim_1, \sim_2, \ldots$ until $\sim_i = \sim_{i+1}$.
- Output $\sim_i$.
- Correctness: Part (3) of the Proposition.
- Termination: Assume the procedure does not terminate. Then, by part (2) of the Proposition, we have an infinite chain

  $$\sim_0 \supset \sim_1 \supset \sim_2 \ldots$$

  This contradicts the finiteness of S.

Partition refinement algorithms

- Idea: think of $\sim$ not as a set of pairs, but as a set of equivalence classes.

Partition refinement algorithms

- Idea: think of $\sim$ not as a set of pairs, but as a set of equivalence classes.
- Recall that $\sim$ is an equivalence relation

Partition refinement algorithms

- Idea: think of $\sim$ not as a set of pairs, but as a set of equivalence classes.
- Recall that $\sim$ is an equivalence relation
- Proposition: $\sim$ is the coarsest partition of S satisfying the following property: For every element $\{E_1, \ldots E_k\} \subseteq S$ of the partition, and for every action a:

Partition refinement algorithms

- Idea: think of $\sim$ not as a set of pairs, but as a set of equivalence classes.
- Recall that $\sim$ is an equivalence relation
- Proposition: $\sim$ is the coarsest partition of S satisfying the following property: For every element $\{E_1, \ldots E_k\} \subseteq S$ of the partition, and for every action a:
  - either none of $E_1, \ldots E_k$ can do an a, or,

Partition refinement algorithms

- Idea: think of $\sim$ not as a set of pairs, but as a set of equivalence classes.
- Recall that $\sim$ is an equivalence relation
- Proposition: $\sim$ is the coarsest partition of S satisfying the following property: For every element $\{E_1, \dots E_k\} \subseteq S$ of the partition, and for every action a:
  - either none of $E_1, \dots E_k$ can do an a, or,
  - all of $E_1, \dots E_k$ can do an a, and there are processes $F_1, \dots, F_k$ such that $E_i \xrightarrow{a} F_i$ for every $1 \leq i \leq k$, and moreover $\{F_1, \dots F_k\}$ is included in an element of the partition.

Partition refinement algorithms

- Idea: think of $\sim$ not as a set of pairs, but as a set of equivalence classes.
- Recall that $\sim$ is an equivalence relation
- Proposition: $\sim$ is the coarsest partition of S satisfying the following property: For every element $\{E_1, \ldots E_k\} \subseteq S$ of the partition, and for every action a:
  - either none of $E_1, \ldots E_k$ can do an a, or,
  - all of $E_1, \ldots E_k$ can do an a, and there are processes $F_1, \ldots, F_k$ such that $E_i \xrightarrow{a} F_i$ for every $1 \leq i \leq k$, and moreover $\{F_1, \ldots F_k\}$ is included in an element of the partition.
- Proof sketch: Show that the elements of a partition satisfy this property if and only if they are the equivalence classes of a bisimulation.

  Show that the coarsest partition corresponds to $\sim$.

Splitting

Given two elements P_1, P_2 of a partition of S and an action a, the result of splitting P_1 w.r.t P_2 and a are the sets

$$\begin{aligned} P_1' &= \{E \in P_1 \mid E \xrightarrow{a} F \text{ for some } F \in P_2\} \\ P_1'' &= P_1 \setminus P_1' \end{aligned}$$

Splitting

Given two elements P_1, P_2 of a partition of S and an action a, the result of splitting P_1 w.r.t P_2 and a are the sets

$$
\begin{array}{rcl}
P_1' & = & \{E \in P_1 \mid E \xrightarrow{a} F \text{ for some } F \in P_2 \ \} \\
P_1'' & = & P_1 \setminus P_1'
\end{array}
$$

Input: S
Output: equivalence classes of $\sim$ on S

Initialize $\Pi := \{S\}$;

Iterate: Choose an action a and $P_1, P_2 \in \Pi$
Split P_1 with respect to P_2 and a;

$\Pi = (\Pi \setminus \{P_1\}) \cup \{P_1', P_1''\}$;
until a fixpoint is reached;

return Π

Complexity

- There are at most $|S| - 1$ splittings.

Complexity

- There are at most $|S| - 1$ splittings.
- Each splitting can be performed in time $O(|S| + |\delta|)$, where δ is set of transitions

Complexity

- There are at most $|S| - 1$ splittings.
- Each splitting can be performed in time $O(|S| + |\delta|)$, where δ is set of transitions
- So the running time is $O(|S| \cdot (|S| + |\delta|)$

Complexity

- There are at most $|S| - 1$ splittings.
- Each splitting can be performed in time $O(|S| + |\delta|)$, where δ is set of transitions
- So the running time is $O(|S| \cdot (|S| + |\delta|)$
- Best known algorithm: $O(|\delta| \cdot log(|S|))$

Complexity

- There are at most $|S| - 1$ splittings.
- Each splitting can be performed in time $O(|S| + |\delta|)$, where δ is set of transitions
- So the running time is $O(|S| \cdot (|S| + |\delta|)$
- Best known algorithm: $O(|\delta| \cdot log(|S|))$
- (Compare deciding language equivalence; which is PSPACE complete)

A Scheduler

Problem: assume n tasks when $n > 1$.

a_i initiates the ith task and b_i signals its completion

The scheduler plans the order of task initiation, ensuring

- actions $a_1 \dots a_n$ carried out cyclically and tasks may terminate in any order
- but a task can not be restarted until its previous operation has finished.
  (a_i and b_i happen alternately for each i.)

More complex temporal properties. Not expressible in CTL* ("not first order" but are "regular").

Expressible using fixed points

Modal Logic+

Z ranges over propositional variables

$\Phi ::= Z \mid \texttt{tt} \mid \texttt{ff} \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid [a]\Phi \mid \langle a \rangle \Phi$

- $\models$ refined to $\models_{\mathsf{V}}$ where V is a **valuation** that assigns a set of states $\mathsf{V}(X)$ to each variable X

$$E \models_{\mathsf{V}} X \text{ iff } E \in \mathsf{V}(X)$$

- $\|\Phi\|$ refined too: $\|\Phi\|_{\mathsf{V}} = \{E : E \models_{\mathsf{V}} \Phi\}$
- $\mathsf{V}[S/X]$ is valuation V' like V except $\mathsf{V}'(X) = S$.

Modal Logic+ II

Proposition The function $\lambda x.\| \Phi \|_{V[x/X]}$ is monotonic for any modal Φ.

- If $\neg$ explicitly in logic then above not true: $\neg X$: $\lambda x. -x$ not monotonic.
  However, define when Φ is **positive** in X: if X occurs within an even number of negations in Φ
  Proposition If Φ is positive in X then $\lambda x.\| \Phi \|_{V[x/X]}$ is monotonic.
- Property given by least fixed point of $\lambda x.\| \Phi \|_{V[x/X]}$ is written $\mu X.\Phi$.
- Property given by greatest fixed point of $\lambda x.\| \Phi \|_{V[x/X]}$ is written $\nu X.\Phi$.

Alternative basis for temporal logic: **modal logic + fixed points**

Modal μ-calculus

Syntax

$\Phi ::= \texttt{tt} \mid \texttt{ff} \mid Z \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid [a]\Phi \mid \langle a \rangle \Phi \mid \nu Z.\,\Phi \mid \mu Z.\,\Phi$

- let σ range over the set $\{\mu, \nu\}$.
- An occurrence of Z is free within Φ if it is not within the scope of an occurrence of σZ. σZ in $\sigma Z.\Phi$ binds free occurrences of Z in Φ.
- Formulas may have multiple fixed points: $\nu Z.\,\mu Y.\,([b]Y \wedge [-]Z)$
- σZ may bind more than one occurrence of Z: $\nu Z.\,\langle \texttt{tick} \rangle Z \wedge \langle \texttt{tock} \rangle Z.$

Semantics

$$
\begin{array}{lll}
E \models_{\mathsf{V}} \mathtt{tt} & & \\
E \not\models_{\mathsf{V}} \mathtt{ff} & & \\
E \models_{\mathsf{V}} Z & \text{iff} & E \in \mathsf{V}(Z) \\
E \models_{\mathsf{V}} \Phi \wedge \Psi & \text{iff} & E \models_{\mathsf{V}} \Phi \text{ and } E \models_{\mathsf{V}} \Psi \\
E \models_{\mathsf{V}} \Phi \vee \Psi & \text{iff} & E \models_{\mathsf{V}} \Phi \text{ or } E \models_{\mathsf{V}} \Psi \\
E \models_{\mathsf{V}} [a]\Phi & \text{iff} & \forall F.\ \text{if } E \xrightarrow{a} F \text{ then } F \models_{\mathsf{V}} \Phi \\
E \models_{\mathsf{V}} \langle a \rangle\Phi & \text{iff} & \exists F. E \xrightarrow{a} F \text{ and } F \models_{\mathsf{V}} \Phi \\
E \models_{\mathsf{V}} \nu Z.\Phi & \text{iff} & E \in \bigcup \{S \ :\ S \subseteq \| \Phi \|_{\mathsf{V}[S/Z]}\} \\
E \models_{\mathsf{V}} \mu Z.\Phi & \text{iff} & E \in \bigcap \{S \ :\ \| \Phi \|_{\mathsf{V}[S/Z]} \subseteq S\}
\end{array}
$$

If f is monotonic (w.r.t $\subseteq$) then $\bigcap\{S : f(S) \subseteq S\}$ is **least** fixed point and $\bigcup\{S : S \subseteq f(S)\}$ is **greatest** fixed point of f.

Semantics II

A slightly different presentation of the clauses for the fixed points dispenses with explicit use of sets $\| \Phi \|_{V}$.

$$
\begin{array}{lll}
E \models_{V} \nu Z.\, \Phi & \text{iff} & \exists S.\, E \in S \text{ and } \forall F \in S.\, F \models_{V[S/Z]} \Phi \\
E \models_{V} \mu Z.\, \Phi & \text{iff} & \forall S.\, \text{if } E \notin S \text{ then } \exists F \notin S.\, F \models_{V[S/Z]} \Phi
\end{array}
$$

Looks second-order because of quantification over sets. Better: $1\frac{1}{2}$-order

If Φ does not contain free variables omit index V: $E \models \Phi$

Unfolding

- An unfolding of $\sigma Z.\,\Phi$ is $\Phi\{\sigma Z.\Phi/Z\}$
  Unfolding of $\nu Z.\,\langle - \rangle Z$ is $\langle - \rangle(\nu Z.\,\langle - \rangle Z)$.
- **Proposition** $E \models_{\mathsf{V}} \sigma Z.\Phi$ iff $E \models_{\mathsf{V}} \Phi\{\sigma Z.\Phi/Z\}$.

Expressiveness I

Modal μ-calculus contains LTL, CTL, CTL*

It also contains Propositional Dynamic Logic (PDL). PDL is modal logic when there is some structure on labels A: closed under operations $+$, ; and $*$

$$
\begin{array}{lll}
E \xrightarrow{w+v} F & \text{iff} & E \xrightarrow{w} F \text{ or } E \xrightarrow{v} F \\
E \xrightarrow{w;v} F & \text{iff} & E \xrightarrow{w} E_1 \xrightarrow{v} F \text{ for some } E_1 \\
E \xrightarrow{w^*} F & \text{iff} & E = F \text{ or } E \xrightarrow{w} E_1 \xrightarrow{w} \dots \xrightarrow{w} E_n \xrightarrow{w} F \text{ for some} \\
 & & n \geq 0 \text{ and } E_1, \dots, E_n
\end{array}
$$

Modal μ-calculus characterisation of bisimulation

$E \equiv F$ if for all closed modal μ-calculus formulas Φ, $E \models \Phi$ iff $F \models \Phi$.

- Theorem: If $E \sim F$ then $E \equiv F$

Modal μ-calculus characterisation of bisimulation

$E \equiv F$ if for all closed modal μ-calculus formulas Φ, $E \models \Phi$ iff $F \models \Phi$.

- Theorem: If $E \sim F$ then $E \equiv F$
- Theorem: If E, F image-finite and $E \equiv F$, then $E \sim F$

Modal μ-calculus characterisation of bisimulation

$E \equiv F$ if for all closed modal μ-calculus formulas Φ, $E \models \Phi$ iff $F \models \Phi$.

- Theorem: If $E \sim F$ then $E \equiv F$
- Theorem: If E, F image-finite and $E \equiv F$, then $E \sim F$
- Alternative perspective: properties
- Let $\|\phi\| = \{E \mid E \models \phi\}$
  (May restrict to particular transition system)

Modal μ-calculus characterisation of bisimulation

$E \equiv F$ if for all closed modal μ-calculus formulas Φ, $E \models \Phi$ iff $F \models \Phi$.

- Theorem: If $E \sim F$ then $E \equiv F$
- Theorem: If E, F image-finite and $E \equiv F$, then $E \sim F$
- Alternative perspective: properties
- Let $\|\phi\| = \{E \mid E \models \phi\}$
  (May restrict to particular transition system)
- First theorem equivalent to properties expressed by modal μ-calculus formulas are bisimulation invariant: if $E \in \|\phi\|$ and $E \sim F$ then $F \in \|\phi\|$

Extend Van Benthem's theorem

- Theorem A FOL formula $\phi(x)$ is equivalent to a modal formula iff $\phi(x)$ is bisimulation invariant.

Extend Van Benthem's theorem

- Theorem A FOL formula $\phi(x)$ is equivalent to a modal formula iff $\phi(x)$ is bisimulation invariant.
- Modal μ-calculus can express properties that are beyond first order logic (such as reachability)

Monadic second order logic (MSO)

$$\phi \ ::= xE_a y \ | \ x = y \ | \ X(x) \ | \ \neg\phi \ | \ \phi_1 \vee \phi_2 \ | \ \exists x.\phi \ | \ \exists X.\phi$$

- $x, y \in Var$ (variables); E_a is binary transition relation for each action a

Monadic second order logic (MSO)

$$\phi ::= xE_a y \mid x = y \mid X(x) \mid \neg \phi \mid \phi_1 \vee \phi_2 \mid \exists x.\phi \mid \exists X.\phi$$

- $x, y \in Var$ (variables); E_a is binary transition relation for each action a
- X ranges over monadic predicate variables VAR; $\exists X$ quantifies over these variables

Monadic second order logic (MSO)

$$\phi ::= xE_a y \mid x = y \mid X(x) \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \exists x.\phi \mid \exists X.\phi$$

- $x, y \in Var$ (variables); E_a is binary transition relation for each action a
- X ranges over monadic predicate variables VAR; $\exists X$ quantifies over these variables
- formulas are interpreted over transition systems
- Valuation $\sigma : Var \to S \cup VAR \to 2^S$ (2^S set of subsets of the processes)

Monadic second order logic (MSO)

$$\phi \ ::= xE_a y \mid x = y \mid X(x) \mid \neg\,\phi \mid \phi_1 \vee \phi_2 \mid \exists x.\phi \mid \exists X.\phi$$

- $x, y \in \mathit{Var}$ (variables); E_a is binary transition relation for each action a
- X ranges over monadic predicate variables VAR; $\exists X$ quantifies over these variables
- formulas are interpreted over transition systems
- Valuation $\sigma : \mathit{Var} \to S \cup \mathit{VAR} \to 2^S$ (2^S set of subsets of the processes)
- $\sigma\{P_1/x_1, \ldots, P_n/x_n, S_1/X_1, \ldots, S_m/X_m\}$ is the valuation that is the same as σ except that its value for x_i is P_i, and for X_j is S_j, $1 \leq i \leq n$, $1 \leq j \leq m$.

Semantics

Inductively define when MSO formula ϕ is true on an LTS with respect to a valuation σ as $\sigma \models \phi$

$$
\begin{array}{lll}
\sigma \models xE_ay & \text{iff} & \sigma(x) \xrightarrow{a} \sigma(y) \\
\sigma \models x = y & \text{iff} & \sigma(x) = \sigma(y) \\
\sigma \models X(x) & \text{iff} & \sigma(x) \in \sigma(X) \\
\sigma \models \neg\phi & \text{iff} & \sigma \not\models \phi \\
\sigma \models \phi_1 \vee \phi_2 & \text{iff} & \sigma \models \phi_1 \text{ or } \sigma \models \phi_2 \\
\sigma \models \exists x.\phi & \text{iff} & \sigma\{P/x\} \models \phi \text{ for some } P \in S \\
\sigma \models \exists X.\phi & \text{iff} & \sigma\{S'/X\} \models \phi \text{ for some } S' \subseteq S
\end{array}
$$

$\forall X.\phi = \neg\exists X.\neg\phi$

Translating modal μ-calculus logic into MSO

The MSO translation of modal formula ϕ relative to variable x is $T_x(\phi)$ which is defined inductively

$$
\begin{array}{lcl}
T_x(\mathtt{tt}) & = & x = x \\
T_x(\mathtt{ff}) & = & \neg(x = x) \\
T_x(\phi_1 \wedge \phi_2) & = & T_x(\phi_1) \wedge T_x(\phi_2) \\
T_x(\phi_1 \vee \phi_2) & = & T_x(\phi_1) \vee T_x(\phi_2) \\
T_x([a]\phi) & = & \forall y.\neg(xE_a y) \vee T_y(\phi) \\
T_x(\langle a \rangle \phi) & = & \exists y.xE_a y \wedge T_y(\phi) \\
T_x(\mu X.\phi) & = & \forall X.(\forall y.(T_y(\phi) \to X(y)) \to X(x)
\end{array}
$$

Translating modal μ-calculus logic into MSO

The MSO translation of modal formula ϕ relative to variable x is $T_x(\phi)$ which is defined inductively

$$
\begin{array}{lcl}
T_x(\mathtt{tt}) & = & x = x \\
T_x(\mathtt{ff}) & = & \neg(x = x) \\
T_x(\phi_1 \wedge \phi_2) & = & T_x(\phi_1) \wedge T_x(\phi_2) \\
T_x(\phi_1 \vee \phi_2) & = & T_x(\phi_1) \vee T_x(\phi_2) \\
T_x([a]\phi) & = & \forall y.\neg(xE_a y) \vee T_y(\phi) \\
T_x(\langle a\rangle\phi) & = & \exists y.xE_a y \wedge T_y(\phi) \\
T_x(\mu X.\phi) & = & \forall X.(\forall y.(T_y(\phi) \rightarrow X(y)) \rightarrow X(x)
\end{array}
$$

Theorem $P \models \phi$ iff $\sigma\{P/x\} \models T_x(\phi)$

Theorem Any closed MSO formula $T_x(\phi)$ is bisimulation invariant

Translating modal μ-calculus logic into MSO

The MSO translation of modal formula ϕ relative to variable x is $T_x(\phi)$ which is defined inductively

$$
\begin{array}{lcl}
T_x(\mathtt{tt}) & = & x = x \\
T_x(\mathtt{ff}) & = & \neg(x = x) \\
T_x(\phi_1 \wedge \phi_2) & = & T_x(\phi_1) \wedge T_x(\phi_2) \\
T_x(\phi_1 \vee \phi_2) & = & T_x(\phi_1) \vee T_x(\phi_2) \\
T_x([a]\phi) & = & \forall y.\neg(xE_a y) \vee T_y(\phi) \\
T_x(\langle a\rangle\phi) & = & \exists y.xE_a y \wedge T_y(\phi) \\
T_x(\mu X.\phi) & = & \forall X.(\forall y.(T_y(\phi) \rightarrow X(y)) \rightarrow X(x)
\end{array}
$$

Theorem $P \models \phi$ iff $\sigma\{P/x\} \models T_x(\phi)$
Theorem Any closed MSO formula $T_x(\phi)$ is bisimulation invariant
A MSO formula $\phi(x)$ is equivalent to closed modal μ-calculus ϕ' provided that for any LTS and for any state P, $\sigma\{P/x\} \models \phi$ iff $P \models \phi'$

Janin and Walukiewicz's theorem

- Theorem A MSO formula $\phi(x)$ is equivalent to a modal μ-calculus formula iff $\phi(x)$ is bisimulation invariant.

Janin and Walukiewicz's theorem

- Theorem A MSO formula $\phi(x)$ is equivalent to a modal μ-calculus formula iff $\phi(x)$ is bisimulation invariant.
- Proof Uses games and automata; see notes

Janin and Walukiewicz's theorem

- Theorem A MSO formula $\phi(x)$ is equivalent to a modal μ-calculus formula iff $\phi(x)$ is bisimulation invariant.
- Proof Uses games and automata; see notes
- Introduce games next time in a model checking setting