Logics, automata, and behavioural properties of discrete event systems

André Arnold

WHAT

- Definition of a (finite) discrete event system P
- Definition of a finite or infinite object $\mathrm{Beh}(\mathrm{P})$ representing the “behaviour” of P
- Definition of a logic $\mathcal{L}$ whose formulas F express properties of $\mathrm{Beh}(\mathrm{P})$

WHAT

- Definition of a (finite) discrete event system P
- Definition of a finite or infinite object $\mathrm{Beh}(\mathrm{P})$ representing the "behaviour" of P
- Definition of a logic $\mathcal{L}$ whose formulas F express properties of $\mathrm{Beh}(\mathrm{P})$

Several possibilities for each definition

WHY

Depending on the choice of Beh and $\mathcal{L}$

- Model-checking Given P and F in $\mathcal{L}$,
  does $\mathrm{Beh}(\mathrm{P})$ has property F? (complexity)

- Satisfiability Given F,
  does there exist P such $\mathrm{Beh}(\mathrm{P})$ has property F? (decidability, complexity)

- Synthesis Given F,
  find a P such $\mathrm{Beh}(\mathrm{P})$ has property F (if any) (complexity)

HOW

I. Logics for linear behaviour

II. Logics for branching behaviour

Transition systems

Let $\mathcal{A}$ be a set of events, and Λ be a set of labels

A transition system (or discrete event systems) is a tuple $P = \langle S, s_\star, T, \lambda \rangle$ where

- S is a finite set of states
- $s_\star \in S$ is the initial state
- $T \subseteq S \times \mathcal{A} \times S$ is the set of transitions
- $\lambda : S \to \Lambda$ is a labelling mapping

Transition systems

Let A be a set of events, and Λ be a set of labels

A transition system (or discrete event systems) is a tuple $P = \langle S, s_\star, T, \lambda \rangle$ where

- S is a finite set of states
- $s_\star \in S$ is the initial state
- $T \subseteq S \times A \times S$ is the set of transitions
- $\lambda : S \to \Lambda$ is a labelling mapping

Remark Usually, $\Lambda = \mathcal{P}(\mathrm{Prop})$ for some set Prop of local properties (possibly empty!), but nothing forbids $\Lambda = S$ and $\lambda = \mathrm{id}_S$.

Part I

Linear behaviour

A path of $P = \langle S, s_\star, T, \lambda \rangle$ is an *infinite* sequence $s_0 a_0 s_1 a_1 \cdots s_n a_n s_{n+1} a_{n+1} \cdots$ (with $s_i \in S$ and $a_i \in A$) such that

- $s_0 = s_\star$,
- $\forall n \in \mathbb{N}, (s_n, a_n, s_{n+1}) \in T$

The trace of the path $s_0 a_0 s_1 a_1 \cdots s_n a_n s_{n+1} a_{n+1} \cdots$ is the sequence

- $a_0 a_1 \cdots a_n a_{n+1} \cdots \in A^\omega$ (event trace)
- $\lambda(s_0)\lambda(s_1)\cdots\lambda(s_n)\lambda(s_{n+1})\cdots \in \Lambda^\omega$ (label trace)
- $(\lambda(s_0), a_0)(\lambda(s_1), a_1)\cdots(\lambda(s_n), a_n)(\lambda(s_{n+1}), a_{n+1})\cdots \in (\Lambda \times A)^\omega$ (full trace)

Linear behaviour

A path of $P = \langle S, s_\star, T, \lambda \rangle$ is an *infinite* sequence $s_0 a_0 s_1 a_1 \cdots s_n a_n s_{n+1} a_{n+1} \cdots$ (with $s_i \in S$ and $a_i \in A$) such that

- $s_0 = s_\star$,
- $\forall n \in \mathbb{N}, (s_n, a_n, s_{n+1}) \in T$

The trace of the path $s_0 a_0 s_1 a_1 \cdots s_n a_n s_{n+1} a_{n+1} \cdots$ is the sequence

- $a_0 a_1 \cdots a_n a_{n+1} \cdots \in A^\omega$ (event trace)
- $\lambda(s_0)\lambda(s_1)\cdots\lambda(s_n)\lambda(s_{n+1})\cdots \in \Lambda^\omega$ (label trace)
- $(\lambda(s_0), a_0)(\lambda(s_1), a_1)\cdots(\lambda(s_n), a_n)(\lambda(s_{n+1}), a_{n+1})\cdots \in (\Lambda \times A)^\omega$ (full trace)

Given a type of trace, $Beh(P)$ is the set of traces of all paths in P

Unlabelling states

Without loss of generality, we may consider only event traces of state-unlabelled transition systems.

$P = \langle S, s_\star, T, \lambda \rangle$ over A and Λ $\quad\rightarrow\quad$ $P' = \langle S, s_\star, T', \lambda' \rangle$ over B and $\mathcal{P}(\emptyset)$

where

- label trace
  - $B = \Lambda$
  - $T' = \{(s, \lambda(s), s') \mid \exists a \in A : (s, a, s') \in T\}$
- full trace
  - $B = \Lambda \times A$
  - $T' = \{(s, (\lambda(s), a), s') \mid (s, a, s') \in T\}$

Specifications

$\mathrm{Beh}(\mathrm{P})$ is a subset of B^{ω} where $\mathrm{B} = A, \Lambda, \Lambda \times A$ according to the type of trace.

Specifications

$\mathrm{Beh}(P)$ is a subset of B^ω where $B = A, \Lambda, \Lambda \times A$ according to the type of trace.

A specification $Spec$ is a subset of B^ω

$$\begin{array}{lll} \mathrm{Beh}(P) \text{ satisfies } Spec & \text{iff} & \mathrm{Beh}(P) \subseteq Spec \\ & \text{iff} & \mathrm{Beh}(P) \cap (B^\omega - Spec) = \emptyset. \end{array}$$

Specifications

$Beh(P)$ is a subset of B^ω where $B = A, \Lambda, \Lambda \times A$ according to the type of trace.

A specification $Spec$ is a subset of B^ω

$Beh(P)$ satisfies $Spec$ iff $Beh(P) \subseteq Spec$

iff $Beh(P) \cap (B^\omega - Spec) = \emptyset$.

$\Rightarrow$ Definition of subsets of B^ω

Büchi automata

A Büchi automaton $\mathcal{A}$ over A is a pair $(\langle Q, q_\star, \Delta\rangle, Q_F)$ where $\langle Q, q_\star, \Delta\rangle$ is an unlabelled transition system and Q_F is a subset of Q.

An infinite word $u = a_0 a_1 \cdots$ is recognized by $\mathcal{A}$ if it is the trace of an accepting path $\pi = q_\star a_0 q_1 a_1 q_2 \cdots$, i.e., which contains infinitely many q_i in Q_F.

$L(\mathcal{A}) \subseteq A^\omega$ is the set of all words recognized by $\mathcal{A}$.

Recognizable sets

A set $L \subseteq A^\omega$ is recognizable if there is a $\mathcal{A}$ such that $L = L(\mathcal{A})$.

Closure properties

- If L and L' are recognizable subsets of A^ω then $L \cup L'$ and $L \cap L'$ are recognizable.
- If L is recognizable then $B^\omega - L$ is recognizable.
- Let $\pi : A \to B$. Let $\pi(L) = \{\pi(a_0)\pi(a_1)\cdots \mid a_0 a_1 \cdots \in L\} \subseteq B^\omega$. If L is recognizable then $\pi(L)$ is recognizable.

Recognizable sets

A set $L \subseteq A^\omega$ is recognizable if there is a $\mathcal{A}$ such that $L = L(\mathcal{A})$.

Closure properties

- If L and L' are recognizable subsets of A^ω then $L \cup L'$ and $L \cap L'$ are recognizable.
- If L is recognizable then $B^\omega - L$ is recognizable.
- Let $\pi : A \to B$. Let $\pi(L) = \{\pi(a_0)\pi(a_1)\cdots \mid a_0 a_1 \cdots \in L\} \subseteq B^\omega$. If L is recognizable then $\pi(L)$ is recognizable.

Proof

If $L = L(\mathcal{A})$ [and $L' = L(\mathcal{A}')$] one can construct $\mathcal{A}''$ such that $L(\mathcal{A}'')$ is equal to

what is needed.

Satisfiability and model-checking

Proposition

$L(\mathcal{A})$ is not empty iff $\mathcal{A}$ contains a state q_1 reachable from $q_\star$ and a cycle $q_1 a_1 q_2 \cdots q_n a_n q_1$ which contains a state $q_j \in Q_F$.

Decidable in linear time (Tarjan's algorithm for strongly connected components)

Satisfiability and model-checking

Proposition

$L(\mathcal{A})$ is not empty iff $\mathcal{A}$ contains a state q_1 reachable from $q_\star$ and a cycle $q_1 a_1 q_2 \cdots q_n a_n q_1$ which contains a state $q_j \in Q_F$.

Decidable in linear time (Tarjan's algorithm for strongly connected components)

$Beh(P) \cap L(\mathcal{A})$ is recognized by the automaton $\mathcal{B} = (\langle S \times Q, (s_\star, q_\star), \Gamma \rangle, S \times Q_F)$ where

$((s,q),a,(s',q')) \in \Gamma$ iff $(s,a,s') \in T$ and $(q,a,q') \in \Delta$.

Deterministic automata

For any state q, $\Delta(q)$ contains at most one state.

Deterministic automata

For any state q, $\Delta(q)$ contains at most one state.

Not every recognizable language is recognized by a deterministic Büchi automaton

Deterministic automata

For any state q, $\Delta(q)$ contains at most one state.

Not every recognizable language is recognized by a deterministic Büchi automaton

counter example

Let $L = \{a, b\}^* a^\omega$. Let us assume that $\mathcal{A}$ is a deterministic Büchi automaton with n states which recognizes L.
For any (reachable) state q there exist a state q' and a state $q'' \in Q_F$ such that $q \xrightarrow{a^m} q' \xrightarrow{a^i} q'' \xrightarrow{a^j} q'$ with $m + i + j = n$.
It follows that the unique path for $(a^n b)\omega$ is accepting.

Other kinds of automata

Let $P = \langle Q, q_\star, \Delta \rangle$ be a transition system and let $\pi = q_\star a_0 q_1 a_1 q_2 \cdots$ be a path. Let $\mathrm{Inf}(\pi)$ be the set of all states ocuring infinitely often in π.

Büchi automaton (P, F) where $F \subseteq Q$. π is accepting if $\mathrm{Inf}(\pi) \cap F$ not empty

Muller automaton $(P, \mathcal{F})$ where $\mathcal{F} \subseteq \mathcal{P}(Q)$. π is accepting if $\mathrm{Inf}(\pi) \in \mathcal{F}$.

Parity automaton (P, ρ) where $\rho : Q \to \mathbb{N}$. π is accepting if $\max\{\rho(q) \mid q \in \mathrm{Inf}(\pi)\}$ is even.

Other kinds of automata

Let $P = \langle Q, q_\star, \Delta \rangle$ be a transition system and let $\pi = q_\star a_0 q_1 a_1 q_2 \cdots$ be a path. Let $\mathrm{Inf}(\pi)$ be the set of all states ocuring infinitely often in π.

Büchi automaton (P, F) where $F \subseteq Q$. π is accepting if $\mathrm{Inf}(\pi) \cap F$ not empty

Muller automaton $(P, \mathcal{F})$ where $\mathcal{F} \subseteq \mathcal{P}(Q)$. π is accepting if $\mathrm{Inf}(\pi) \in \mathcal{F}$.

Parity automaton (P, ρ) where $\rho : Q \to \mathbb{N}$. π is accepting if $\max\{\rho(q) \mid q \in \mathrm{Inf}(\pi)\}$ is even.

Büchi to parity: $\rho(q) = \begin{cases} 2 & \text{if } q \in F \\ 1 & \text{otherwise} \end{cases}$

Determinisation of automata

Theorem[McNaughton, 1966] and many others since then.

Every recognizable set is recognized by a deterministic Muller automaton and by a deterministic parity automaton.

Determinisation of automata

Theorem[McNaughton, 1966] and many others since then.

Every recognizable set is recognized by a deterministic Muller automaton and by a deterministic parity automaton.

Example

Let $A = \{a_0, a_1, a_2\}$ and let $L = A^* a_0^\omega \cup (A^* a_2)^\omega$.

$Q = \{q_0, q_1, q_2\}, \qquad q_* = q_0, \qquad \rho(q_i) = i.$

$\forall i, j, \Delta(q_i, a_j) = \{q_j\}.$

From automata to logic

Let $\mathcal{A}$= $(\langle Q, q_\star, \Delta\rangle, Q_F)$ be a Büchi automaton and let $u \in A^\omega$ seen as a mapping $u : \mathbb{N} \to A$ (i.e., $u = u(0)u(1)\cdots u(n)\cdots$).
u is recognized by $\mathcal{A}$ iff with each $q \in Q$ is associated a subset E_q of $\mathbb{N}$ such that

- these subsets form a partition of $\mathbb{N}$, more precisely
  - they are pairwise disjoint: $\forall q, q' \in Q, q \neq q' \Rightarrow E_q \cap E_{q'} = \emptyset$
  - they cover $\mathbb{N}$: $\mathbb{N} \subseteq \bigcup_{q\in Q} E_q$
- $0 \in E_{q_\star}$
- for any $n \in \mathbb{N}$ there exists $(q, a, q') \in \Delta$ such that $n \in E_q$, $u(n) = a$, and $n + 1 \in E_{q'}$
- there exists $q \in Q_F$ such that E_q is infinite (i.e., $\forall n \in \mathbb{N}, \exists m \in E_q : n \leq m$)

From automata to logic

Let $\mathcal{A}$= $(\langle Q, q_\star, \Delta\rangle, Q_F)$ be a Büchi automaton and let $u \in A^\omega$ seen as a mapping $u : \mathbb{N} \to A$ (i.e., $u = u(0)u(1)\cdots u(n)\cdots$).
u is recognized by $\mathcal{A}$ iff with each $q \in Q$ is associated a subset E_q of $\mathbb{N}$ such that

- these subsets form a partition of $\mathbb{N}$, more precisely
  - they are pairwise disjoint: $\forall q, q' \in Q, q \neq q' \Rightarrow E_q \cap E_{q'} = \emptyset$
  - they cover $\mathbb{N}$: $\mathbb{N} \subseteq \bigcup_{q\in Q} E_q$
- $0 \in E_{q_\star}$
- for any $n \in \mathbb{N}$ there exists $(q, a, q') \in \Delta$ such that $n \in E_q$, $u(n) = a$, and $n + 1 \in E_{q'}$
- there exists $q \in Q_F$ such that E_q is infinite (i.e., $\forall n \in \mathbb{N}, \exists m \in E_q : n \leq m$)

Trivia $n + 1 \in E$ iff $\exists m \in E : n \leq m$ and $\forall k \in \mathbb{N}(k \leq n$ or $m \leq k)$

$0 \in E$ iff $\exists m \in E : \forall k \in \mathbb{N}, m \leq k$

Monadic second order logic

Let Var_0 be a set of individual variables and Var_1 be a set of set variables. For each $a \in A$ let V_a be a unary predicate.
The formulas are defined inductively by

- $V_a(x)$ with $x \in Var_0$ and $a \in A$
- $x \leq y$, $x \in X$ with $x, y \in Var_0$ and $X \in Var_1$,
- $F \vee F'$, $F \wedge F'$, $\neg F$, with F and F' are formulas.
- $\exists x F$, $\forall x F$, $\exists X F$, $\forall X F$, with $x \in Var_0$, $X \in Var_1$, and F a formula.

Monadic second order logic

Let Var_0 be a set of individual variables and Var_1 be a set of set variables. For each $a \in A$ let V_a be a unary predicate.
The formulas are defined inductively by

- $V_a(x)$ with $x \in Var_0$ and $a \in A$
- $x \leq y$, $x \in X$ with $x, y \in Var_0$ and $X \in Var_1$,
- $F \vee F'$, $F \wedge F'$, $\neg F$, with F and F' are formulas.
- $\exists x F$, $\forall x F$, $\exists X F$, $\forall X F$, with $x \in Var_0$, $X \in Var_1$, and F a formula.

Let $F(x, x', \ldots, X, X', \ldots)$ be a formula whose free variables are $x, x', \ldots$(individual) and $X, X', \ldots$ (set). Let u be a word.
Let $n, n', \ldots$ (resp., $E, E', \ldots$) be natural numbers (resp. sets) associated with the free individual (resp. set) variables of F.
We define (by induction) the satisfaction relation $u \models F(n, n', \ldots, E, E', \ldots)$ which means that $F(n, n', \ldots, E, E' \ldots)$ is true in u by

- $u \models V_a(n)$ iff $u(n) = a$
- $\cdots\cdots$ straightforward!

MSOL definability

A set $L \subseteq A^\omega$ is MSOL-definable if there is a closed formula F such that $L = \{u \mid u \models F\}$

Theorem[Büchi, 1960]
A set L is recognizable iff it is MSOL definable.

Proof
$\Rightarrow$ see above
$\Leftarrow$ by induction, using the closure properties given above

First-order definabiliy

Every FOL-definable language is MSOL-definable.

The converse is not true! Example $\{u \in \{a,b\}^\omega \mid u(n) = a \Rightarrow n \textit{ is even}\}$.

Theorem[Kamp, 1968]

A language is FOL-definable iff it is LTL-definable

LTL

$$\begin{aligned} F \quad ::= \quad & true | false | a | \neg a | b | \neg b | \cdots | \\ & F \vee F | F \wedge F | \neg F | \\ & \mathcal{N} F | \mathcal{A} F | F \, \mathcal{U} F \end{aligned}$$

Definition of $u \models F$.
Let $u[i]$ be the suffix $u(i)u(i+1)\cdots$ of u.

- $u \models true$, $u \not\models false$
- $u \models a$ iff $u(0) = a$.
- $u \models F \vee F'$ (resp. $\wedge$) iff $u \models F$ or (resp. and) $u \models F'$
- $u \models \mathcal{N} F$ iff $u[1] \models F$,
- $u \models \mathcal{A} F$ iff $\forall i, u[i] \models F$,
- $u \models F \mathcal{U} F'$ iff there exists i such that
  - $u[i] \models F'$ and $\forall j, 0 \leq j < i \Rightarrow u[j] \models F$

Duality

Extension of De Morgan's law

- $\neg \mathcal{N} \mathrm{F} \equiv \mathcal{N} \neg \mathrm{F}$
- $\neg \mathcal{A} \mathrm{F} \equiv \mathrm{true} \, \mathcal{U} \neg \mathrm{F}$
- $\neg (\mathrm{F} \, \mathcal{U} \, \mathrm{F}') \equiv (\mathcal{A} \neg \mathrm{F}') \quad \vee \quad (\neg \mathrm{F}') \, \mathcal{U} (\neg \mathrm{F} \wedge \neg \mathrm{F}')$

LTL to FOL

By induction on F in LTL : there exists $\widehat{F}(x)$ in FOL such that

$$\forall i, (u[i] \models F \Leftrightarrow u \models \widehat{F}(i))$$

- $\widehat{a} = V_a(x)$,
- $\widehat{\mathcal{N}F} = \widehat{F}(x+1)$,
- $\widehat{\mathcal{A}F} = \forall y, (x \leq y \Rightarrow \widehat{F}(y))$,
- $\widehat{F\mathcal{U}F'} = \exists y : \quad x \leq y \wedge \widehat{F'}(y) \wedge \forall z (x \geq z < y \Rightarrow \widehat{F}(z))$.

Fixed points in LTL

With a formula F we associate the defined language $[\![\mathrm{F}]\!] = \{u \mid u \models \mathrm{F}\}$.

- $[\![\mathcal{A}\mathrm{F}]\!]$ is the greatest language L such that $\mathrm{L} = [\![\mathrm{F}]\!] \cap \mathrm{AL}$.
- $[\![\mathrm{F}\,\mathcal{U}\,\mathrm{F}']\!]$ is the least language L such that $\mathrm{L} = [\![\mathrm{F}']\!] \cup ([\![\mathrm{F}]\!] \cap \mathrm{AL})$.

Fixed points in LTL

With a formula F we associate the defined language $[\![F]\!] = \{u \mid u \models F\}$.

- $[\![\mathcal{A}F]\!]$ is the greatest language L such that $L = [\![F]\!] \cap AL$.
- $[\![F \mathcal{U} F']\!]$ is the least language L such that $L = [\![F']\!] \cup ([\![F]\!] \cap AL)$.

Notation

- $[\![\mathcal{A}F]\!] = \nu L.([\![F]\!] \cap AL)$.
- $[\![F \mathcal{U} F']\!] = \mu L.([\![F']\!] \cup ([\![F]\!] \cap AL))$.

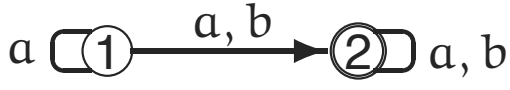
Alternating automata

$[\![a \mathcal{U} b]\!] = a^* b \{a, b\}^\omega$ is recognized by

$q_1 \xrightarrow{a} q_1, \quad q_1 \xrightarrow{b} q_2, \quad q_2 \xrightarrow{a} q_2, \quad q_2 \xrightarrow{b} q_2,$

$[\![\mathcal{A}(a \mathcal{U} b)]\!] = (a^* b)^\omega$ is recognized by adding

$q_0 \xrightarrow{a} q_0 \wedge q_1, \quad q_0 \xrightarrow{b} q_0 \wedge q_1,$ with initial state $q_0 \wedge q_1$.



Alternating automata

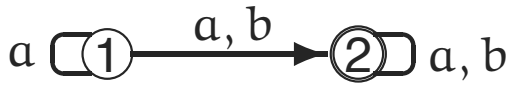
$[\![a\,\mathcal{U}\,b]\!] = a^*b\{a,b\}^\omega$ is recognized by

$q_1 \xrightarrow{a} q_1, \quad q_1 \xrightarrow{b} q_2, \quad q_2 \xrightarrow{a} q_2, \quad q_2 \xrightarrow{b} q_2,$

$[\![\mathcal{A}(a\,\mathcal{U}\,b)]\!] = (a^*b)^\omega$ is recognized by adding

$q_0 \xrightarrow{a} q_0 \wedge q_1, \quad q_0 \xrightarrow{b} q_0 \wedge q_1,$ with initial state $q_0 \wedge q_1$.

How to get a nondeterministic automaton (possibly deterministic)



Alternating automata

$[\![a \mathcal{U} b]\!] = a^* b \{a, b\}^\omega$ is recognized by

$q_1 \xrightarrow{a} q_1, \quad q_1 \xrightarrow{b} q_2, \quad q_2 \xrightarrow{a} q_2, \quad q_2 \xrightarrow{b} q_2,$

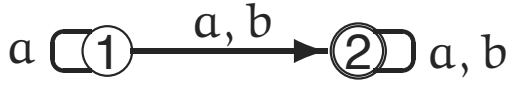
$[\![\mathcal{A}(a \mathcal{U} b)]\!] = (a^* b)^\omega$ is recognized by adding

$q_0 \xrightarrow{a} q_0 \wedge q_1, \quad q_0 \xrightarrow{b} q_0 \wedge q_1,$ with initial state $q_0 \wedge q_1$.

The usual powerset construction does not work!

$\{q_0, q_1\} \xrightarrow{a} \{q_0, q_1\}, \quad \{q_0, q_1\} \xrightarrow{b} \{q_0, q_1, q_2\},$

$\{q_0, q_1, q_2\} \xrightarrow{a} \{q_0, q_1, q_2\}, \quad \{q_0, q_1, q_2\} \xrightarrow{b} \{q_0, q_1, q_2\},$



Alternating automata

$[\![a \mathcal{U} b]\!] = a^* b \{a, b\}^\omega$ is recognized by

$q_1 \xrightarrow{a} q_1, \quad q_1 \xrightarrow{b} q_2, \quad q_2 \xrightarrow{a} q_2, \quad q_2 \xrightarrow{b} q_2,$

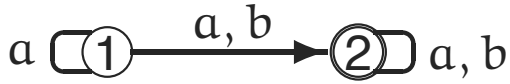
$[\![\mathcal{A}(a \mathcal{U} b)]\!] = (a^* b)^\omega$ is recognized by adding

$q_0 \xrightarrow{a} q_0 \wedge q_1, \quad q_0 \xrightarrow{b} q_0 \wedge q_1,$ with initial state $q_0 \wedge q_1$.

The usual powerset construction does not work!

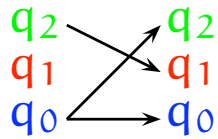
$\{q_0, q_1\} \xrightarrow{a} \{q_0, q_1\}, \quad \{q_0, q_1\} \xrightarrow{b} \{q_0, q_1, q_2\},$

$\{q_0, q_1, q_2\} \xrightarrow{a} \{q_0, q_1, q_2\}, \quad \{q_0, q_1, q_2\} \xrightarrow{b} \{q_0, q_1, q_2\},$

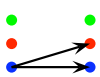
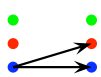


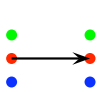
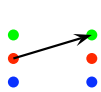
If b^ω is recognized then ba^ω is recognized as well

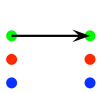
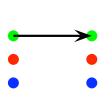
Histories

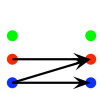
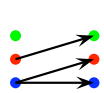
Let $\mathcal{R}$ be the set of binary relations over $\{q_0, q_1, q_2\}$. Example: $R =$ 

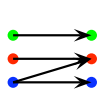
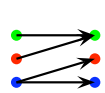
Let us define the language L on $A \times \mathcal{R}$ recognized by

$q_0 \vdash (a,$  $) \to \{q_0, q_1\}, \quad q_0 \vdash (b,$  $) \to q_0, q_1,$

$q_1 \vdash (a,$  $) \to q_1, \quad q_1 \vdash (b,$  $) \to q_2,$

$q_2 \vdash (a,$  $) \to q_2, \quad q_2 \vdash (b,$  $) \to q_2,$

$\{q_0, q_1\} \vdash (a,$  $) \to \{q_0, q_1\}, \quad \{q_0, q_1\} \vdash (b,$  $) \to \{q_0, q_1, q_2\},$

$\{q_0, q_1, q_2\} \vdash (a,$  $) \to \{q_0, q_1, q_2\}, \quad \{q_0, q_1, q_2\} \vdash (b,$  $) \to \{q_0, q_1, q_2\},$

where all states are accepting.

Accepting graphs

A graph $G = R_0 R_1 \cdots \in \mathcal{R}^\omega$ is accepting if all its infinite paths satisfy the parity condition.

A word u is recognized by the previous alternating automata (i.e., is in $[\![\mathcal{A}(a\,\mathcal{U} b)]\!] = (a^* b)^\omega$) if there is a word $u \times G \in L$ such that G is accepting.

Accepting graphs

A graph $G = R_0 R_1 \cdots \in \mathcal{R}^\omega$ is accepting if all its infinite paths satisfy the parity condition.

A word u is recognized by the previous alternating automata (i.e., is in $[\![\mathcal{A}(a\,\mathcal{U} b)]\!] = (a^* b)^\omega$) if there is a word $u \times G \in L$ such that G is accepting.

NB. This is indeed the formal definition of a word recognized by an alternating automaton.

Accepting graphs

A graph $G = R_0 R_1 \cdots \in \mathcal{R}^\omega$ is accepting if all its infinite paths satisfy the parity condition.

A word u is recognized by the previous alternating automata (i.e., is in $[\![\mathcal{A}(a\,\mathcal{U}b)]\!] = (a^*b)^\omega$) if there is a word $u \times G \in L$ such that G is accepting.

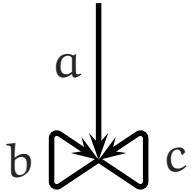
NB. This is indeed the formal definition of a word recognized by an alternating automaton.

By McNauhton's theorem the set $\mathcal{G}$ of all accepting G is recognized by a deterministic parity automaton.

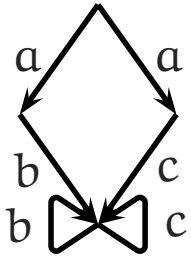
It follows that the language $\{(u, G) \mid (u, G) \in L, G \in \mathcal{G}\}$ is recognized by a parity automaton (the product of the automata recognizing L and $\mathcal{G}$).

Part II

Branching properties

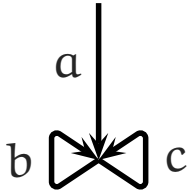


and

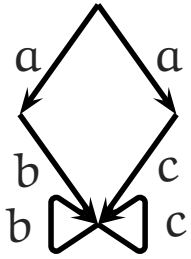


have the same linear behaviour: $a\{b, c\}^{\omega}$

Branching properties



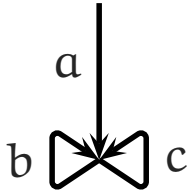
and



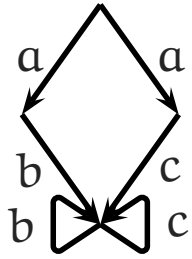
have the same linear behaviour: $a\{b, c\}^{\omega}$

A branching property In every state where b is firable, c is firable too.

Branching properties



and

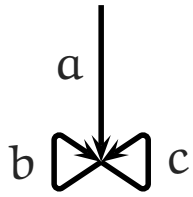


have the same linear behaviour: $a\{b, c\}^\omega$

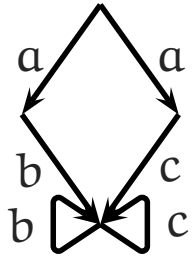
YES NO

A branching property In every state where b is firable, c is firable too.

Branching properties



and



have the same linear behaviour: $a\{b,c\}^\omega$

YES NO

A branching property In every state where b is firable, c is firable too.

A minimal deterministic transition system P is fully determined by its linear behaviour $L(P)$. The above property can be expressed by

$$\forall u \in A^*, \quad (\exists w \in A^\omega : ubw \in L(P)) \Rightarrow (\exists w' \in A^\omega : ucw' \in L(P))$$

which has not the linear form $\forall u \in L(P), u \in Spec$

Transition systems and monotonic functions

Let $\mathrm{P} = \langle S, s_\star, \mathrm{T} \rangle$ be a state-unlabelled transition system.

Let $\mathcal{M}(S)$ be the set of all monotonic functions over $\mathcal{P}(S)$

Transition systems and monotonic functions

Let $P = \langle S, s_\star, T \rangle$ be a state-unlabelled transition system.

Let $\mathcal{M}(S)$ be the set of all monotonic functions over $\mathcal{P}(S)$

For every $a \in A$, $\mathcal{M}(S)$ contains the two functions $\langle a \rangle_P$ and $[a]_P$ from $\mathcal{P}(S)$ to $\mathcal{P}(S)$ defined by

- $\langle a \rangle_P(E) = \{s \in S \mid \exists (s, a, s') \in T : s' \in E\}$
- $[a]_P(E) = \{s \in S \mid \forall (s, a, s') \in T, s' \in E\}$

Duality: $S - \langle a \rangle_P(E) = [a]_P(S - E)$

Transition systems and monotonic functions

Let $P = \langle S, s_\star, T \rangle$ be a state-unlabelled transition system.

Let $\mathcal{M}(S)$ be the set of all monotonic functions over $\mathcal{P}(S)$

For every $a \in A$, $\mathcal{M}(S)$ contains the two functions $\langle a \rangle_P$ and $[a]_P$ from $\mathcal{P}(S)$ to $\mathcal{P}(S)$ defined by

- $\langle a \rangle_P(E) = \{s \in S \mid \exists (s, a, s') \in T : s' \in E\}$
- $[a]_P(E) = \{s \in S \mid \forall (s, a, s') \in T, s' \in E\}$

Duality: $S - \langle a \rangle_P(E) = [a]_P(S - E)$

Back to the exemple : $[b]_P(\emptyset) \cup (\langle b \rangle_P(S) \wedge \langle c \rangle_P(S)) = S$

Event CTL

Syntax

$$
\begin{array}{rcll}
F & ::= & true | false | F \vee F | F \wedge F | & \\
 & & \langle a \rangle F | [a] F | & a \in A \\
 & & \langle \mathcal{A} F \rangle | [\mathcal{A} F] | \langle F \, \mathcal{U} F \rangle | [F \, \mathcal{U} F] &
\end{array}
$$

Event CTL

Syntax

$$\begin{array}{rcll} F & ::= & true|false|F \vee F|F \wedge F| & \\ & & \langle a \rangle F|[a]F| & a \in A \\ & & \langle \mathcal{A} F \rangle|[\mathcal{A} F]|\langle F \mathcal{U} F \rangle|[F \mathcal{U} F] & \end{array}$$

Semantics $[\![F]\!]_P \subseteq S$ defined by induction on F

- $[\![\langle a \rangle F]\!]_P = \langle a \rangle_P([\![F]\!]_P), \quad [\![[a]F]\!]_P = [a]_P([\![F]\!]_P),$
- $[\![\langle \mathcal{A} F \rangle]\!]_P = \nu X.([\![F]\!]_P \wedge (\bigvee_{a \in A} \langle a \rangle_P(X))),$
  $[\![[\mathcal{A} F]]\!]_P = \nu X.([\![F]\!]_P \wedge (\bigwedge_{a \in A} [a]_P(X))),$
- $[\![\langle F \mathcal{U} F' \rangle]\!]_P = \mu X.([\![F']\!]_P \vee ([\![F]\!]_P \wedge \bigvee_{a \in A} \langle a \rangle_P X)),$
  $[\![[F \mathcal{U} F']]\!]_P = \mu X.([\![F']\!]_P \vee ([\![F]\!]_P \wedge \bigwedge_{a \in A} [a]_P X))$

Event CTL

Syntax

$$F ::= \mathrm{true} \mid \mathrm{false} \mid F \vee F \mid F \wedge F \mid \langle a \rangle F \mid [a] F \mid \quad a \in \mathcal{A}$$
$$\langle \mathcal{A} F \rangle \mid [\mathcal{A} F] \mid \langle F \mathcal{U} F \rangle \mid [F \mathcal{U} F]$$

Semantics $[\![F]\!]_P \subseteq S$ defined by induction on F

- $[\![\langle a \rangle F]\!]_P = \langle a \rangle_P([\![F]\!]_P), \quad [\![[a]F]\!]_P = [a]_P([\![F]\!]_P),$
- $[\![\langle \mathcal{A} F \rangle]\!]_P = \nu X.([\![F]\!]_P \wedge (\bigvee_{a \in \mathcal{A}} \langle a \rangle_P(X))),$
  $[\![[\mathcal{A} F]]\!]_P = \nu X.([\![F]\!]_P \wedge (\bigwedge_{a \in \mathcal{A}} [a]_P(X))),$
- $[\![\langle F \mathcal{U} F' \rangle]\!]_P = \mu X.([\![F']\!]_P \vee ([\![F]\!]_P \wedge \bigvee_{a \in \mathcal{A}} \langle a \rangle_P X)),$
  $[\![[F \mathcal{U} F']]\!]_P = \mu X.([\![F']\!]_P \vee ([\![F]\!]_P \wedge \bigwedge_{a \in \mathcal{A}} [a]_P X))$

$$P \models F \text{ iff } s_\star \in [\![F]\!]_P$$

Example (and counter-example)

After an $\mathtt{a}$, there will always be a $\mathtt{b}$: $[\mathtt{a}]\mathtt{false} \vee \langle \mathtt{a} \rangle [\mathtt{true} \mathcal{U} \langle \mathtt{b} \rangle \mathtt{true}]$

Example (and counter-example)

After an $\mathtt{a}$, there will always be a $\mathtt{b}$: $[\mathtt{a}]\mathtt{false} \vee \langle \mathtt{a} \rangle [\mathtt{true} \mathcal{U} \langle \mathtt{b} \rangle \mathtt{true}]$

After an $\mathtt{a}$, there will always be infinitely many $\mathtt{b}$'s

Example (and counter-example)

After an a, there will always be a b: $[a]false \vee \langle a \rangle [true \mathcal{U} \langle b \rangle true]$

After an a, there will always be infinitely many b's

$f(X,Y) = \langle b \rangle_P(Y) \cup \bigcap_{c \neq b} [c]_P(X)$ from $\mathcal{P}(S) \times \mathcal{P}(S)$ to $\mathcal{P}(S)$ in $\mathcal{M}(P)$,

$g(Y) = \mu X.f(X,Y)$,
$h = \nu Y.g(Y)$,
$[a]_P(\emptyset) \cup \langle a \rangle_P(h)$

Modal parity automata

$\mathcal{A} = \langle Q, q_\star, \Delta, \rho \rangle$ with

- $\rho : Q \to \mathbb{N}$
- $\Delta : Q \to \mathcal{P}(\mathcal{C})$ where
- $\mathcal{C} = \mathcal{P}(Q \cup \{\langle a \rangle q, [a]q \mid a \in A, q \in Q\})$

Modal parity automata

$\mathcal{A} = \langle Q, q_\star, \Delta, \rho \rangle$ with

- $\rho : Q \to \mathbb{N}$
- $\Delta : Q \to \mathcal{P}(\mathcal{C})$ where
- $\mathcal{C} = \mathcal{P}(Q \cup \{\langle a \rangle q, [a]q \mid a \in A, q \in Q\})$

Exemple: there will always be infinitely many b's
$f(X,Y) = \langle b \rangle_P(Y) \cup \bigcap_{c \neq b} [c]_P(X), \quad g(Y) = \mu X.f(X,Y), \quad h = \nu Y.g(Y)$

$Q = \{q_X, q_Y\}$, $q_\star = q_Y$, $\rho(q_X) = 1$, $\rho(q_Y) = 2$,
$\Delta(q_X) = \{\{\langle b \rangle q_Y\}, \{[c]q_X \mid c \neq b\}\}$ (to be read $\langle b \rangle q_Y \vee (\bigwedge_{c \neq b} [c]q_X)$)
$\Delta(q_Y) = \{\{q_X\}\}$ (to be read q_X)

Semantics

$$[\![\mathcal{A}]\!]_P \subseteq \mathcal{P}(S)$$

$$P \models \mathcal{A} \Leftrightarrow s_\star \in [\![\mathcal{A}]\!]_P$$

Semantics

$$[\![\mathcal{A}]\!]_P \subseteq \mathcal{P}(S)$$

$$P \models \mathcal{A} \Leftrightarrow s_\star \in [\![\mathcal{A}]\!]_P$$

How to define (compute) $[\![\mathcal{A}]\!]_P$?

Semantics

$$[\![\mathcal{A}]\!]_P \subseteq \mathcal{P}(S)$$

$$P \models \mathcal{A} \Leftrightarrow s_\star \in [\![\mathcal{A}]\!]_P$$

How to define (compute) $[\![\mathcal{A}]\!]_P$? Using parity games or the μ-calculus

Semantics

$$[\![\mathcal{A}]\!]_P \subseteq \mathcal{P}(S)$$

$$P \models \mathcal{A} \Leftrightarrow s_\star \in [\![\mathcal{A}]\!]_P$$

How to define (compute) $[\![\mathcal{A}]\!]_P$? Using parity games or the μ-calculus

$[\![\mathcal{A}]\!]_{\mathcal{P}}$ is the component of index $q_\star$ of the solution $\{E_q \mid q \in Q\}$ of a system of fixed-point equations $\Sigma(\mathcal{A}, P)$.

Systems of equations

Let $\mathcal{A}$ whose set of states is $Q = \{q_1, \ldots, q_n\}$ such that $i < j \Rightarrow \rho(q_i) \leq \rho(q_j)$.
With any P we associate the system of n fixed-point equations $\Sigma(\mathcal{A}, P)$:

$$\left.\begin{array}{lcl} X_1 & \stackrel{\theta_1}{=} & f_1(X_1, \ldots, X_n) \\ \vdots & & \\ X_i & \stackrel{\theta_i}{=} & f_i(X_1, \ldots, X_n) \\ \vdots & & \\ X_n & \stackrel{\theta_n}{=} & f_n(X_1, \ldots, X_n) \end{array}\right\} \text{ where } \theta_i = \mu \text{ if } i \text{ is odd, } \nu \text{ if } i \text{ is even}$$

and $f_i(X_1, \ldots, X_n)$ is the monotonic function from $\mathcal{P}(S)^n$ to $\mathcal{P}(S)$ obtained by substituting in $\Delta(q_i)$

- $\cup$ for $\vee$, and $\cap$ for $\wedge$,
- X_j for q_j
- $\langle a \rangle_P(X_j)$ for $\langle a \rangle q_j$, and $[a]_P(X_j)$ for $[a]q_j$

Solving a system of fixed-point equations

Computation (by induction on n) of the solution $\mathrm{Sol}(\Sigma) \subseteq \mathcal{P}(S)^n$ of

$$\Sigma = \begin{cases} X_1 & \overset{\theta_1}{=} & f_1(X_1, X_2, \ldots, X_n) \\ X_2 & \overset{\theta_2}{=} & f_2(X_1, X_2, \ldots, X_n) \\ \vdots & & \\ X_n & \overset{\theta_n}{=} & f_n(X_1, X_2, \ldots, X_n) \end{cases}$$

Solving a system of fixed-point equations

Computation (by induction on n) of the solution $Sol(\Sigma) \subseteq \mathcal{P}(S)^n$ of

$$\Sigma = \begin{cases} X_1 & \overset{\theta_1}{=} & f_1(X_1, X_2, \ldots, X_n) \\ X_2 & \overset{\theta_2}{=} & f_2(X_1, X_2, \ldots, X_n) \\ \vdots & & \\ X_n & \overset{\theta_n}{=} & f_n(X_1, X_2, \ldots, X_n) \end{cases}$$

Compute the monotonic function

$g_1(X_2, \ldots, X_n) = \theta_1 X_1 . f_1(X_1, X_2, \ldots, X_n) \in \mathcal{P}(S)^{n-1} \to \mathcal{P}(S)$

Solving a system of fixed-point equations

Computation (by induction on n) of the solution $Sol(\Sigma) \subseteq \mathcal{P}(S)^n$ of

$$\Sigma = \begin{cases} X_1 & \overset{\theta_1}{=} & f_1(X_1, X_2, \ldots, X_n) \\ X_2 & \overset{\theta_2}{=} & f_2(X_1, X_2, \ldots, X_n) \\ \vdots & & \\ X_n & \overset{\theta_n}{=} & f_n(X_1, X_2, \ldots, X_n) \end{cases}$$

Compute the monotonic function

$g_1(X_2, \ldots, X_n) = \theta_1 X_1 . f_1(X_1, X_2, \ldots, X_n) \in \mathcal{P}(S)^{n-1} \to \mathcal{P}(S)$

Compute the solution $\{E_2, \ldots E_n\}$ of Σ'

$$\Sigma' = \begin{cases} X_2 & \overset{\theta_2}{=} & f_2(g_1(X_2, \ldots, X_n), X_2, \ldots, X_n) \\ \vdots & & \\ X_n & \overset{\theta_n}{=} & f_n(g_1(X_2, \ldots, X_n), X_2, , \ldots, X_n) \end{cases}$$

Solving a system of fixed-point equations

Computation (by induction on n) of the solution $Sol(\Sigma) \subseteq \mathcal{P}(S)^n$ of

$$\Sigma = \begin{cases} X_1 & \overset{\theta_1}{=} & f_1(X_1, X_2, \ldots, X_n) \\ X_2 & \overset{\theta_2}{=} & f_2(X_1, X_2, \ldots, X_n) \\ \vdots & & \\ X_n & \overset{\theta_n}{=} & f_n(X_1, X_2, \ldots, X_n) \end{cases}$$

Compute the monotonic function
$g_1(X_2, \ldots, X_n) = \theta_1 X_1 . f_1(X_1, X_2, \ldots, X_n) \in \mathcal{P}(S)^{n-1} \to \mathcal{P}(S)$

Compute the solution $\{E_2, \ldots E_n\}$ of Σ'

The solution of Σ is $\{g_1(E_2, \ldots, E_n), E_2, \ldots, E_n\}$

The modal μ-calculus

Syntax

$$t ::= true | false | X | t \vee t | t \wedge t | \langle a \rangle t | [a] t | \mu X.t | \nu X.t$$

Semantics

For any transition system P, for any term t and for any sequence $X_1, \ldots X_n$ which contains all the free variables of t we define by induction the monotonic function $[\![t]\!]_P(X_1, \ldots, X_n) : \mathcal{P}(S)^n \to \mathcal{P}(S)$.
Note: if t is closed then $[\![t]\!]_P() \subseteq S$.

The modal μ-calculus

Syntax

$$t ::= true | false | X | t \vee t | t \wedge t | \langle a \rangle t | [a] t | \mu X.t | \nu X.t$$

Semantics

For any transition system P, for any term t and for any sequence $X_1, \ldots X_n$ which contains all the free variables of t we define by induction the monotonic function $[\![t]\!]_P(X_1, \ldots, X_n) : \mathcal{P}(S)^n \to \mathcal{P}(S)$.
Note: if t is closed then $[\![t]\!]_P() \subseteq S$.

- if $t = true$ (resp $false$) then $[\![t]\!]_P(E_1, \ldots, E_n) = S$ (resp. $\emptyset$)
- if $t = X_i$ then $[\![t]\!]_P(E_1, \ldots, E_n) = E_i$
- if $t = t_1 \vee t_2$ (resp. $\wedge$) then $[\![t]\!]_P(E_1, \ldots, E_n) = [\![t_1]\!]_P(E_1, \ldots, E_n) \cup [\![t_2]\!]_P(E_1, \ldots, E_n)$ (resp. $\cap$)
- if $t = \langle a \rangle t'$ (resp $[a]$) then $[\![t]\!](E_1, \ldots, E_n) = \langle a \rangle_P([\![t']\!]_P(E_1, \ldots, E_n))$ (resp. $[a]_P$)
- if $t = \theta X.t'$ then $[\![t]\!]_P(E_1, \ldots, E_n) = \theta X.[\![t']\!]_P(X, E_1, \ldots, E_n)$.

Parity automata and μ-calculus

Proposition For any automaton $\mathcal{A}$ there exists a μ-term $t_{\mathcal{A}}$ such that for any P, $[\![\mathcal{A}]\!]_P = [\![t_{\mathcal{A}}]\!]()$.

Parity automata and μ-calculus

Proposition For any automaton $\mathcal{A}$ there exists a μ-term $t_{\mathcal{A}}$ such that for any P, $[\![\mathcal{A}]\!]_P = [\![t_{\mathcal{A}}]\!]()$.

Let $t_i = \Delta(q_i)$ and $f_i(X_1, \ldots X_n) = [\![t_i]\!]_P(X_1, \ldots, X_n)$

$$\Sigma(\mathcal{A}) \begin{cases} X_1 & \stackrel{\theta_1}{=} & t_1 \\ X_2 & \stackrel{\theta_2}{=} & t_2 \\ \vdots & & \\ X_n & \stackrel{\theta_n}{=} & t_n \end{cases} \qquad \Sigma(\mathcal{A}, P) \begin{cases} X_1 & \stackrel{\theta_1}{=} & f_1(X_1, \ldots, X_n) \\ X_2 & \stackrel{\theta_2}{=} & f_2(X_1, \ldots, X_n) \\ \vdots & & \\ X_n & \stackrel{\theta_n}{=} & f_n(X_1, \ldots, X_n) \end{cases}$$

Parity automata and μ-calculus

Proposition For any automaton $\mathcal{A}$ there exists a μ-term $t_{\mathcal{A}}$ such that for any P, $[\![\mathcal{A}]\!]_P = [\![t_{\mathcal{A}}]\!]()$.

Let $t_i = \Delta(q_i)$ and $f_i(X_1, \ldots X_n) = [\![t_i]\!]_P(X_1, \ldots, X_n)$

$$\Sigma(\mathcal{A}) \begin{cases} X_1 & \stackrel{\theta_1}{=} & t_1 \\ X_2 & \stackrel{\theta_2}{=} & t_2 \\ \vdots & & \\ X_n & \stackrel{\theta_n}{=} & t_n \end{cases} \qquad \Sigma(\mathcal{A}, P) \begin{cases} X_1 & \stackrel{\theta_1}{=} & f_1(X_1, \ldots, X_n) \\ X_2 & \stackrel{\theta_2}{=} & f_2(X_1, \ldots, X_n) \\ \vdots & & \\ X_n & \stackrel{\theta_n}{=} & f_n(X_1, \ldots, X_n) \end{cases}$$

$t'_1 = \theta_1 X_1 . t_1 \qquad g_1(X2, \ldots, X_n) = \theta_1 X_1 . f_1(X_1, \ldots, X_n) = [\![t'_1]\!]_P(X_2, \ldots, X_n)$

Parity automata and μ-calculus

Proposition For any automaton $\mathcal{A}$ there exists a μ-term $t_{\mathcal{A}}$ such that for any P, $[\![\mathcal{A}]\!]_P = [\![t_{\mathcal{A}}]\!]()$.

Let $t_i = \Delta(q_i)$ and $f_i(X_1, \ldots X_n) = [\![t_i]\!]_P(X_1, \ldots, X_n)$

$$\Sigma(\mathcal{A}) \begin{cases} X_1 & \stackrel{\theta_1}{=} & t_1 \\ X_2 & \stackrel{\theta_2}{=} & t_2 \\ \vdots & & \\ X_n & \stackrel{\theta_n}{=} & t_n \end{cases} \qquad \Sigma(\mathcal{A}, P) \begin{cases} X_1 & \stackrel{\theta_1}{=} & f_1(X_1, \ldots, X_n) \\ X_2 & \stackrel{\theta_2}{=} & f_2(X_1, \ldots, X_n) \\ \vdots & & \\ X_n & \stackrel{\theta_n}{=} & f_n(X_1, \ldots, X_n) \end{cases}$$

$$t_1' = \theta_1 X_1.t_1 \qquad g_1(X2, \ldots, X_n) = \theta_1 X_1.f_1(X_1, \ldots, X_n) = [\![t_1']\!]_P(X_2, \ldots, X_n)$$

$$\Sigma'(\mathcal{A}) \begin{cases} X_2 & \stackrel{\theta_2}{=} & t_2[X_1 := t_1'] \\ \vdots & & \\ X_n & \stackrel{\theta_n}{=} & t_n[X_1 := t_1'] \end{cases}$$

$$\Sigma'(\mathcal{A}, P) \begin{cases} X_2 & \stackrel{\theta_2}{=} & f_2(g_1(X2, \ldots, X_n), X_2, \ldots, X_n) \\ \vdots & & \\ X_n & \stackrel{\theta_n}{=} & f_n(g_1(X2, \ldots, X_n), X_2, \ldots, X_n) \end{cases}$$

And vice-versa

Proposition For any closed term t there exists an automaton $\mathcal{A}_\mathrm{t}$ such that $[\![\mathrm{t}]\!]_\mathrm{P}() = [\![\mathcal{A}_\mathrm{t}]\!]_\mathrm{P}$ for any P.

And vice-versa

Proposition For any closed term t there exists an automaton $\mathcal{A}_t$ such that $[\![t]\!]_P() = [\![\mathcal{A}_t]\!]_P$ for any P.

An incomplete automaton is an automaton containing some states (say $q_1, \ldots, q_k$) for which ρ and Δ are not defined. (Obviously, $q_\star$ must be defined.) The "syntactic" solution of $\Sigma(\mathcal{A})$ contains the free variables $X_1, \ldots, X_k$. Thus $[\![\mathcal{A}]\!]_P(X_1, \ldots, X_k)$ is a mapping from $\mathcal{P}(S)^k \to \mathcal{P}(S)$.

And vice-versa

Proposition For any closed term t there exists an automaton $\mathcal{A}_t$ such that $[\![t]\!]_P() = [\![\mathcal{A}_t]\!]_P$ for any P.

Lemma For any term t whose free variables are $X_1, \ldots, X_k$, there is an incomplete automaton $\mathcal{A}$ with undefined states $q_1, \ldots, q_k$, such that for any P, $[\![\mathcal{A}]\!]_P(X_1, \ldots, X_k) = [\![t]\!]_P(X_1, \ldots, X_k)$.

And vice-versa

Proposition For any closed term t there exists an automaton $\mathcal{A}_t$ such that $[\![t]\!]_P() = [\![\mathcal{A}_t]\!]_P$ for any P.

Lemma For any term t whose free variables are $X_1, \ldots, X_k$, there is an incomplete automaton $\mathcal{A}$ with undefined states $q_1, \ldots, q_k$, such that for any P, $[\![\mathcal{A}]\!]_P(X_1, \ldots, X_k) = [\![t]\!]_P(X_1, \ldots, X_k)$.

Proof by induction. Let $\mathcal{A}^{(i)}$ be "equivalent" to $t_i(X_1^{(i)}, \ldots, X_{k_i}^{(i)})$.

- The automaton equivalent to $\mu X_1^{(i)}.t_1$ is obtained by taking $q_1^{(i)}$ as initial state and defining it by $\rho(q_1^{(i)})$ equal to any odd number greater than $\rho(q_{k+1}^{(i)}), \ldots, \rho(q_{k+n}^{(i)})$, and $\Delta(q_1^{(i)}) = \{\{q_\star^{(i)}\}\}$.
- The automaton equivalent to $t_1 \vee t_2$ (resp. $t_1 \wedge t_2$) is obtained by adding to the "disjoint" union of $\mathcal{A}^{(1)}$ and $A^{(2)}$ the new initial state $q_\star$ of rank 0 defined by $\Delta(q_\star) = \{\{q_\star^{(1)}\}, \{q_\star^{(2)}\}\}$ (resp. $\Delta(q_\star) = \{\{q_\star^{(1)}, q_\star^{(2)}\}\}$)

MSOL-definability

Basic predicates $V_a(x, y)$: $V_a(s, s')$ is true in P if $(s, a, s') \in T$

MSOL-definability

Basic predicates $V_a(x, y)$: $V_a(s, s')$ is true in P if $(s, a, s') \in T$

Proposition For any μ- term t, whose free variables are $\{X_1, \ldots, X_n\}$, there exists a formula $F_t(z, X_1, \ldots, X_n)$ in MSOL such that $F_t(s, E_1, \ldots, E_n)$ is true in P iff $s \in [\![t]\!]_P(E_1, \ldots, E_n)$

MSOL-definability

Basic predicates $V_a(x, y)$: $V_a(s, s')$ is true in P if $(s, a, s') \in T$

Proposition For any μ- term t, whose free variables are $\{X_1, \ldots, X_n\}$, there exists a formula $F_t(z, X_1, \ldots, X_n)$ in MSOL such that $F_t(s, E_1, \ldots, E_n)$ is true in P iff $s \in [\![t]\!]_P(E_1, \ldots, E_n)$

Let $G_t(Z, X_1, \ldots, X_n)$ be equal to $\forall z \in Z, F_t(z, X_1, \ldots, X_n)$
(so that $G_t(E, E_1, \ldots E_n)$ true in P iff $E = [\![t]\!]_P(E_1, \ldots, E_n)$)

- if $t = \langle a \rangle X$ then $F_t(z, X) = \exists x \in X : V_a(z, x)$
- if $t = \mu X_1.t'$ then $F_t(z, X_2, \ldots, X_n) =$
  $\exists Z : z \in Z \wedge G_{t'}(Z, Z, X_2, \ldots, X_n) \wedge \forall X(G_{t'}(X, X, X_2, \ldots, X_n) \Rightarrow Z \subseteq X)$.
- etc.

And the converse

is false

$$F(X) = \forall x(x \in X \Longrightarrow V_a(x, x))$$

In $P_1 = (s_\star \xrightarrow{a} s_\star)$, $F(E)$ is true iff $E = \{s_\star\}$

In $P_2 = (s_\star \xrightarrow{a} s \xrightarrow{a} s_\star)$, $F(E)$ is true iff $E = \emptyset$

And the converse

is false

$$F(X) = \forall x(x \in X \Longrightarrow V_a(x, x))$$

In $P_1 = (s_\star \xrightarrow{a} s_\star)$, $F(E)$ is true iff $E = \{s_\star\}$

In $P_2 = (s_\star \xrightarrow{a} s \xrightarrow{a} s_\star)$, $F(E)$ is true iff $E = \emptyset$

For any closed μ-term t, $[\![t]\!]_{P_1} = \emptyset$ iff $[\![t]\!]_{P_2} = \emptyset$

And the converse

is false

$$F(X) = \forall x(x \in X \Longrightarrow V_a(x, x))$$

In $P_1 = (s_\star \xrightarrow{a} s_\star)$, $F(E)$ is true iff $E = \{s_\star\}$
In $P_2 = (s_\star \xrightarrow{a} s \xrightarrow{a} s_\star)$, $F(E)$ is true iff $E = \emptyset$

For any closed μ-term t, $[\![t]\!]_{P_1} = \emptyset$ iff $[\![t]\!]_{P_2} = \emptyset$

For any subset E of $S_1 = \{s_\star\}$, let E' be the subset of $S_2 = \{s_\star, s\}$ such that $E' = \emptyset$ if $E = \emptyset$ and $E' = S_2$ if $E = S_1$.
Then for any t, $E = [\![t]\!]_{P_1}(E_1, \ldots, E_n)$ iff $E' = [\![t]\!]_{P_2}(E'_1, \ldots, E'_n)$.

And the converse

is false

$$F(X) = \forall x(x \in X \Longrightarrow V_a(x, x))$$

In $P_1 = (s_\star \xrightarrow{a} s_\star)$, $F(E)$ is true iff $E = \{s_\star\}$
In $P_2 = (s_\star \xrightarrow{a} s \xrightarrow{a} s_\star)$, $F(E)$ is true iff $E = \emptyset$

For any closed μ-term t, $[\![t]\!]_{P_1} = \emptyset$ iff $[\![t]\!]_{P_2} = \emptyset$

Inductive proof of: For any t, $E = [\![t]\!]_{P_1}(E_1, \ldots, E_n)$ iff $E' = [\![t]\!]_{P_2}(E'_1, \ldots, E'_n)$.

- $\langle a\rangle_{P_1}(\emptyset) = \langle a\rangle_{P_2}(\emptyset) = [a]_{P_1}(\emptyset) = [a]_{P_2}(\emptyset) = \emptyset$
  $\langle a\rangle_{P_1}(S_1) = [a]_{P_1}(S_1) = S_1, \quad \langle a\rangle_{P_2}(S_2) = [a]_{P_2}(S_2) = S_2$
- Let $t = \mu X_1.t'$, let $f_i(X_1, X_2) = [\![t']\!]_{P_i}(X_1, X_2)$ and $g_i(X_2) = [\![t]\!]_{P_i}(X_2)$.
  Let $E_1 = g_1(E_2) = f_1(E_1, E_2)$.
  - If $E_1 = \emptyset = f_1(\emptyset, E_2)$ then $\emptyset = f_2(\emptyset, E'_2)$ hence and $g_2(E'_2) = \emptyset = E'_1$.
  - If $E_1 = S_1$ then $f_1(\emptyset, E_2) = S_1$, hence
    $S_2 = f_2(\emptyset, E'_2) \subseteq f_2(g_2(E'_2), E'_2) = g_2(E'_2)$, hence $g_2(E'_2) = S_2 = E'_1$.

Bisimulation

A bisimulation between P and P' is a relation $R \subseteq S \times S'$ such that

- $R(s_\star, s'_\star)$
- If $R(s, s')$ then
  - $\forall (s, a, s_1) \in T, \exists (s', a, s'_1) \in T' : R(s_1, s'_1)$,
  - $\forall (s', a, s'_1) \in T', \exists (s, a, s_1) \in T : R(s_1, s'_1)$,

Bisimulation

A bisimulation between P and P' is a relation $R \subseteq S \times S'$ such that

- $R(s_\star, s'_\star)$
- If $R(s, s')$ then
  - $\forall (s, a, s_1) \in T, \exists (s', a, s'_1) \in T' : R(s_1, s'_1)$,
  - $\forall (s', a, s'_1) \in T', \exists (s, a, s_1) \in T : R(s_1, s'_1)$,

$R = \{(s_\star, s_\star), (s_\star, s)\}$ is a bisimulation between $P_1 = (s_\star \xrightarrow{a} s_\star)$ and $P_2 = (s_\star \xrightarrow{a} s \xrightarrow{a} s_\star)$.

Bisimulation

A bisimulation between P and P' is a relation $R \subseteq S \times S'$ such that

- $R(s_\star, s'_\star)$
- If $R(s, s')$ then
  - $\forall (s, a, s_1) \in T, \exists (s', a, s'_1) \in T' : R(s_1, s'_1)$,
  - $\forall (s', a, s'_1) \in T', \exists (s, a, s_1) \in T : R(s_1, s'_1)$,

$R = \{(s_\star, s_\star), (s_\star, s)\}$ is a bisimulation between $P_1 = (s_\star \xrightarrow{a} s_\star)$ and $P_2 = (s_\star \xrightarrow{a} s \xrightarrow{a} s_\star)$.

Fact

- If R is a bisimulation between P and P' then R^{-1} is a bisimulation between $P'$and P.
- If R is a bisimulation between P and P', and if R' is a bisimulation between P' and P'', then $R \circ R'$ is a bisimulation between P and P''

Saturated sets

Let R be a bisimulation between P and P'. A subset E of S is R-saturated if $R^{-1}(R(E)) = E$. (i.e. if $R(s_1, s')$ and $R(s_2, s')$ then $s_1 \in E \Leftrightarrow s_2 \in E$).

Saturated sets

Let R be a bisimulation between P and P'. A subset E of S is R-saturated if $R^{-1}(R(E)) = E$. (i.e. if $R(s_1, s')$ and $R(s_2, s')$ then $s_1 \in E \Leftrightarrow s_2 \in E$).

Fact

- If E is R-saturated then $R(E)$ is R^{-1}-saturated.
- $\emptyset$ and S are R-saturated.
- if E_1 and E_2 are R-saturated then $E_1 \cup E_2$, $E_1 \cap E_2$, and $S - E_1$ are R-saturated.

Bisimulation invariance

Proposition Let P and P'. If there is a bisimulation between P and P' then for any (closed) automaton $\mathcal{A}$, $\quad P \models \mathcal{A} \leftrightarrow P' \models \mathcal{A}$.

Lemma Let R be a bisimulation between P and P'.
For any μ-term t and any R-saturated subsets $E_1, \ldots, E_n$,

- the set $E = [\![t]\!]_P(E_1, \ldots, E_n)$ is R-saturated.
- $R(E) = [\![t]\!]_{P'}(R(E_1), \ldots, R(E_n))$.

Corollary $[\![\mathcal{A}]\!]_P$ is R-saturated and $[\![\mathcal{A}]\!]_{P'} = R([\![\mathcal{A}]\!]_P)$ hence $[\![\mathcal{A}]\!]_P = R^{-1}([\![\mathcal{A}]\!]_{P'})$.

Infinite transitions systems

All the previous definitions ($[\![\mathcal{A}]\!]_P$, $[\![t]\!]_P(X_1, \ldots, X_n)$, bisimulation) and results (parity automata $\Leftrightarrow$ μ-terms $\Rightarrow$ MSOL formulas, bisimulation invariance) are still valid for infinite transition systems.

Infinite transitions systems

All the previous definitions ($[\![\mathcal{A}]\!]_P$, $[\![t]\!]_P(X_1, \ldots, X_n)$, bisimulation) and results (parity automata $\Leftrightarrow$ μ-terms $\Rightarrow$ MSOL formulas, bisimulation invariance) are still valid for infinite transition systems.

Example: P is in bisimulation with its (infinite) tree unfolding $\mathcal{TU}(P)$.

Infinite transitions systems

All the previous definitions ($[\![\mathcal{A}]\!]_P$, $[\![t]\!]_P(X_1, \ldots, X_n)$, bisimulation) and results (parity automata $\Leftrightarrow$ μ-terms $\Rightarrow$ MSOL formulas, bisimulation invariance) are still valid for infinite transition systems.

Example: P is in bisimulation with its (infinite) tree unfolding $\mathcal{TU}(P)$.

Finite model property If $\mathcal{A}$ has a model ($\exists P : P \models \mathcal{A}$) then it has a finite model.

Infinite transitions systems

All the previous definitions ($[\![\mathcal{A}]\!]_P$, $[\![t]\!]_P(X_1, \ldots, X_n)$, bisimulation) and results (parity automata $\Leftrightarrow$ μ-terms $\Rightarrow$ MSOL formulas, bisimulation invariance) are still valid for infinite transition systems.

Example: P is in bisimulation with its (infinite) tree unfolding $\mathcal{TU}(P)$.

Finite model property If $\mathcal{A}$ has a model ($\exists P : P \models \mathcal{A}$) then it has a finite model.

Fact MSOL has NOT the finite model property.

Infinite transitions systems

All the previous definitions ($[\![\mathcal{A}]\!]_P$, $[\![t]\!]_P(X_1, \ldots, X_n)$, bisimulation) and results (parity automata $\Leftrightarrow$ μ-terms $\Rightarrow$ MSOL formulas, bisimulation invariance) are still valid for infinite transition systems.

Example: P is in bisimulation with its (infinite) tree unfolding $TU(P)$.

Finite model property If $\mathcal{A}$ has a model ($\exists P : P \models \mathcal{A}$) then it has a finite model.

Fact MSOL has NOT the finite model property.

$V(x, y) = \bigvee_{a \in A} V_a(x, y)$, $\quad F = \forall x, \exists y : V(x, y) \quad \wedge \quad \forall y((\exists x : V(x, y)) \Rightarrow$
$\forall x, x', (V(x, y) \wedge V(x', y) \Rightarrow x = x'))$
(i.e. each state is of indegree at most 1)

F is true in P iff P is an infinite tree.

MSOL and bisimulation invariance

A MSOL-formula $F(x)$ with one free variable is bisimulation-invariant if

for any P, P', any bisimulation R between P and P' and any $(s, s') \in R$ one has:

$F(s)$ is true in P iff $F(s')$ is true in P'

MSOL and bisimulation invariance

A MSOL-formula $F(x)$ with one free variable is bisimulation-invariant if

for any P, P', any bisimulation R between P and P' and any $(s, s') \in R$ one has:

$F(s)$ is true in P iff $F(s')$ is true in P'

Theorem [Janin-Walukiewicz, 1996]

If $F(x)$ is bisimulation-invariant then there exists $\mathcal{A}$ such that for any P and s,

$F(s)$ is true in P iff $s \in [\![\mathcal{A}]\!]_P$.

Satisfiability and the problem of conjunction

A closed μ-term t is satisfiable (denoted by $\models t$) if there is a P such that $P \models t$ (i.e. $s_\star \in [\![t]\!]_P$)

Fact $\models t$ iff $\exists P : [\![t]\!]_P \neq \emptyset$ (One can take any state in $[\![t]\!]_P$ as initial state)

Satisfiability and the problem of conjunction

A closed μ-term t is satisfiable (denoted by $\models t$) if there is a P such that $P \models t$ (i.e. $s_\star \in [\![t]\!]_P$)

Fact $\models t$ iff $\exists P : [\![t]\!]_P \neq \emptyset$ (One can take any state in $[\![t]\!]_P$ as initial state)

Let t and t' be two closed terms.

- $\models t \vee t'$ iff $\models t$ or $\models t'$ (because $[\![t \vee t']\!]_P = [\![t]\!]_P \vee [\![t']\!]_P$)
- $\models t \wedge t'$ implies $\models t$ yet $\models t'$

Satisfiability and the problem of conjunction

A closed μ-term t is satisfiable (denoted by $\models t$) if there is a P such that $P \models t$ (i.e. $s_\star \in [\![t]\!]_P$)

Fact $\models t$ iff $\exists P : [\![t]\!]_P \neq \emptyset$ (One can take any state in $[\![t]\!]_P$ as initial state)

Let t and t' be two closed terms.

- $\models t \vee t'$ iff $\models t$ or $\models t'$ (because $[\![t \vee t']\!]_P = [\![t]\!]_P \vee [\![t']\!]_P$)
- $\models t \wedge t'$ implies $\models t$ yet $\models t'$

The converse is not always true:

$\models \langle a \rangle true$ and $\models [a] false$ but $\not\models \langle a \rangle true \wedge [a] false$

Satisfiability and the problem of conjunction

A closed μ-term t is satisfiable (denoted by $\models t$) if there is a P such that $P \models t$ (i.e. $s_\star \in [\![t]\!]_P$)

Fact $\models t$ iff $\exists P : [\![t]\!]_P \neq \emptyset$ (One can take any state in $[\![t]\!]_P$ as initial state)

Let t and t' be two closed terms.

- $\models t \vee t'$ iff $\models t$ or $\models t'$ (because $[\![t \vee t']\!]_P = [\![t]\!]_P \vee [\![t']\!]_P$)
- $\models t \wedge t'$ implies $\models t$ yet $\models t'$

The converse is not always true:

$\models \langle a \rangle true$ and $\models [a] false$ but $\not\models \langle a \rangle true \wedge [a] false$

and not always false:

$\models \langle a \rangle true$ and $\models [b] false$ but $\models \langle a \rangle true \wedge [b] false$

Guarded μ-terms

Some conjunction are problematic ($\langle a \rangle true \wedge [a] false$),
some are not problematic ($\langle a \rangle true \wedge [a] false$)

Guarded μ-terms

Some conjunction are problematic ($\langle a \rangle true \wedge [a] false$),
some are not problematic ($\langle a \rangle true \wedge [a] false$)

But some are of unknown status: $\nu X.t(\mu Y.(X \wedge t'(X, Y)))$

Guarded μ-terms

Some conjunction are problematic ($\langle a\rangle true \wedge [a] false$),
some are not problematic ($\langle a\rangle true \wedge [a] false$)

But some are of unknown status: $\nu X.t(\mu Y.(X \wedge t'(X, Y)))$

A μ-term is guarded if each occurrence of a variable X appears in a subterm $\langle a\rangle X$ or $[a]X$

Guarded μ-terms

Some conjunction are problematic ($\langle a\rangle true \wedge [a] false$),
some are not problematic ($\langle a\rangle true \wedge [a] false$)

But some are of unknown status: $\nu X.t(\mu Y.(X \wedge t'(X, Y)))$

A μ-term is guarded if each occurrence of a variable X appears in a subterm $\langle a\rangle X$ or $[a]X$

Theorem Each μ-term is (effectively) equivalent to a guarded one

Corollary Each automaton is equivalent to a guarded automaton $\mathcal{A}$
i.e. where $\Delta : Q \to \mathcal{P}(\mathcal{C})$ where $\mathcal{C} = \mathcal{P}(\{\langle a\rangle q, [a]q \mid a \in A, q \in Q\})$ instead of $\mathcal{C} = \mathcal{P}(Q \cup \{\langle a\rangle q, [a]q \mid a \in A, q \in Q\})$

Simulation Theorem

Elimination (in a guarded automaton) of all problematic conjunctions

(powerset construction + histories + MacNaughton)

Simulation Theorem

Elimination (in a guarded automaton) of all problematic conjunctions

(powerset construction + histories + MacNaughton)

Theorem [Janin-Walukiewicz, 1995]

Every automaton is equivalent to an automaton $\mathcal{A}$ such that any conjunction $c \in \Delta(q)$ has the form $\bigwedge_{a \in A} \langle a \rangle q_1 \wedge \cdots \wedge \langle a \rangle q_n \wedge [a](q_1 \vee \cdots \vee q_n)$