Expressiveness of Temporal Logics

Francois Laroussinie and Nicolas Markey
Lab. Specification et Verification
ENS Cachan & CNRS, France
August 3, 2006

Outline

1 BT-temporal logics with Past
2 CTL∗ vs Monadic second order logic
3 Automata theory and BT-temporal logics
4 Alternating-time temporal logic

CTL∗ + Past

Definition

\[ PCTL^* \models \varphi, \psi ::= P_1 | \ldots | \neg \varphi | \varphi \land \psi | E_{\varphi_p} | A_{\varphi_p} \]
\[ PCTL^*_p \models \varphi_p, \psi_p ::= \varphi | \neg \varphi_p | \varphi_p \land \psi_p | X_{\varphi_p} | \varphi_p U \psi_p \]

with \( P \in AP \)

\( PCTL^* \) formulae are interpreted over states with an history.

Structure of the past

In the linear-time case, past and future are symmetric.

In the branching-time case, several choices are possible. Here we consider a past which is:

- **determined**: an history contains the events which already took place. **Ockhamist past**.
  Thus past and future have a different structure.
- **finite**: the studied behavior has a starting point.
- **cumulative**: whenever the system performs some steps, its history becomes richer and longer.

Structure of the past

In the linear-time case, past and future are symmetric.

In the branching-time case, several choices are possible. Here we consider a past which is:

- **determined**: an history contains the events which already took place. **Ockhamist past**.
  Thus past and future have a different structure.
- **finite**: the studied behavior has a starting point.
- **cumulative**: whenever the system performs some steps, its history becomes richer and longer.
Structure of the past

In the linear-time case, past and future are symmetric.
In the branching-time case, several choices are possible.
Here we consider a past which is:
- **determined**: an history contains the events which already took place.
- **Ockhamist past**: thus past and future have a different structure.
- **finite**: the studied behavior has a starting point.
- **cumulative**: whenever the system performs some steps, its history becomes richer and longer.

PCTL* formulas are interpreted over finite prefixes:
- the last state is the current state,
- the other ones define the history.

Structure of the past

In the linear-time case, past and future are symmetric.
In the branching-time case, several choices are possible.
Here we consider a past which is:

Adding S or X□1 (Laroussinie & Schnoebelen, 1994)

L' is initially as expressive as L : ∀ϕ ∈ L, ∃ϕ' ∈ L', such that for any state q in any KS, we have q |= ϕ iff q |= ϕ'.

- **ECTL +** is not as expressive as UB + S ,
  E(a ∨ b U c) U d can be expressed in UB + S
- **ECTL +** is not as expressive as UB + X□1 ,
  EG(a ∨ X a) ≡i EG(a ∨ X□1 a ¬ X□1 t t)

PAST may add expressivity !

Adding S or X□1 (Laroussinie & Schnoebelen, 1994)

∗

L' is initially as expressive as L : ∀ϕ ∈ L, ∃ϕ' ∈ L', such that for any state q in any KS, we have q |= ϕ iff q |= ϕ'.

- **ECTL +** is not as expressive as UB + S ,
  E(a ∨ b U c) U d can be expressed in UB + S
- **ECTL +** is not as expressive as UB + X□1 ,
  EG(a ∨ X a) ≡i EG(a ∨ X□1 a ¬ X□1 t t)

PAST may add expressivity !

Adding S or X□1 (Laroussinie & Schnoebelen, 1994)

L' is initially as expressive as L : ∀ϕ ∈ L, ∃ϕ' ∈ L', such that for any state q in any KS, we have q |= ϕ iff q |= ϕ'.

- **ECTL +** is not as expressive as UB + S ,
  E(a ∨ b U c) U d can be expressed in UB + S
- **ECTL +** is not as expressive as UB + X□1 ,
  EG(a ∨ X a) ≡i EG(a ∨ X□1 a ¬ X□1 t t)

PAST may add expressivity !

Adding S or X□1 (Laroussinie & Schnoebelen, 1994)

L' is initially as expressive as L : ∀ϕ ∈ L, ∃ϕ' ∈ L', such that for any state q in any KS, we have q |= ϕ iff q |= ϕ'.

- **ECTL +** is not as expressive as UB + S ,
  E(a ∨ b U c) U d can be expressed in UB + S
- **ECTL +** is not as expressive as UB + X□1 ,
  EG(a ∨ X a) ≡i EG(a ∨ X□1 a ¬ X□1 t t)

PAST may add expressivity !

Adding S or X□1 (Laroussinie & Schnoebelen, 1994)

L' is initially as expressive as L : ∀ϕ ∈ L, ∃ϕ' ∈ L', such that for any state q in any KS, we have q |= ϕ iff q |= ϕ'.

- **ECTL +** is not as expressive as UB + S ,
  E(a ∨ b U c) U d can be expressed in UB + S
- **ECTL +** is not as expressive as UB + X□1 ,
  EG(a ∨ X a) ≡i EG(a ∨ X□1 a ¬ X□1 t t)

PAST may add expressivity !
Adding $F\Box 1$

$CTL + F\Box 1$ can be weakly separated.

$F\Box 1 EF P$ cannot be (fully) separated.

Definition

A formula is weakly separated when no past-modalities occur in the scope of a future-modality.

Theorem (Laroussinie & Schnoebelen, 1994)

Any $CTL + F\Box 1$ formula can be separated.

Any $B(X, X\Box 1, S)$ formula can be separated.

(based on separation rules of (Gabbay, 1987))

Example of separation

$E(P_1 \land F\Box 1 P_2) U (P_3 \land F\Box 1 P_4) \equiv$

$(P_3 \land F\Box 1 P_4) \lor$

$F\Box 1 P_3 \land \ldots$

$F\Box 1 P_4 \land EP_1 U P_3 \lor$

$EP_1 U (P_1 \land P_2 \land EP_1 U P_3)$

Example of separation

$E(P_1 \land F\Box 1 P_2) U (P_3 \land F\Box 1 P_4) \equiv$

$(P_3 \land F\Box 1 P_4) \lor$

$F\Box 1 P_3 \land \ldots$

$F\Box 1 P_4 \land EP_1 U P_3 \lor$

$EP_1 U (P_1 \land P_4 \land EP_1 U P_3)$
Separation and initial equivalence

If a logic can be weakly separated, it is initially equivalent to its pure-future fragment.

Let $\Phi$ be a weakly separated formula: every past-modality in $\Phi$ occurs at the root of $\Phi$ (possibly in the scope of boolean connectives) or in the scope of another past-modality.

We have:
- $\psi \lor \phi \equiv \psi$ (1)
- $X^1 \psi \equiv \psi \downarrow$ (2)

By applying rules (1) and (2), we can easily deduce that $\Phi$ is initially equivalent to some pure-future formula.

Theorem (Hafer & Thomas, 1987)

$PCTL^*$ is initially equivalent to $CTL^*$.

(based on Kamp’s theorem)

BTL with $F\Box^1$ (Laroussinie & Schnoebelen, 1994)

The following results hold for initial equivalence.
- $B(X)$ is as expressive as $B(X, X^{-1}, S)$.  
- $CTL$ is as expressive as $CTL^+ + F^{-1}$,  
  (but $CTL^+ + F^{-1}$ is exponentially more succinct)
- $ECTL^+$ is as expressive as $ECTL^+ + F^{-1}$.
- $ECTL + F^{-1}$ is strictly more expressive than $ECTL$  
  ($EF(a \land G^{-1} b)$ cannot be expressed in $ECTL$)
- $ECTL + F^{-1}$ is strictly less expressive than $ECTL^+$.  
  ($E(F a \land F b)$ cannot be expressed in $ECTL + F^{-1}$)
Relation with other formalisms

Relationship between linear-time temporal logics and first-order logic or with automata theory is well known.

What about branching-time temporal logics?

Need a formalism able to quantify over paths and not only on positions along a path.

Outline

- BT-temporal logics with Past
- CTL* vs Monadic second order logic
- Automata theory and BT-temporal logics
- Alternating-time temporal logic

Monadic Second Order Logic

Consider the monadic second order logic MSOL \( (\prec, \Sigma) \) to express properties of \( \Sigma \)-labeled trees. It contains:
- individual variables \( x, y, z, \ldots \) (for the nodes)
- set variables \( X, Y, Z, \ldots \) (for set of nodes)
- predicate constants \( P_a \) for \( a \in \Sigma \)
- \( \And x = y, x < y, x \in X, x \in P_a \)
- \( \And \, \Or, \neg, \exists, \forall \)

(FOL \( (\prec, \Sigma) \) is the restriction without set variables.)

The Monadic path logic MPL is the restriction of MSOL where the interpretation of set variables \( X \) ranges only over paths.

Monadic Second Order Logic

Consider the monadic second order logic MSOL \( (\prec, \Sigma) \) to express properties of \( \Sigma \)-labeled trees. It contains:
- individual variables \( x, y, z, \ldots \) (for the nodes)
- set variables \( X, Y, Z, \ldots \) (for set of nodes)
- predicate constants \( P_a \) for \( a \in \Sigma \)
- \( \And x = y, x < y, x \in X, x \in P_a \)
- \( \And \, \Or, \neg, \exists, \forall \)

(FOL \( (\prec, \Sigma) \) is the restriction without set variables.)

The Monadic path logic MPL is the restriction of MSOL where the interpretation of set variables \( X \) ranges only over paths.
Monadic Second Order Logic

Consider the monadic second order logic MSOL (\(<, \Sigma\)) to express properties of \(\Sigma\)-labeled trees. It contains:
- individual variables \(x, y, z, \ldots\) (for the nodes)
- set variables \(X, Y, Z, \ldots\) (for set of nodes)
- predicate constants \(P_a\) for \(a \in \Sigma\)
- And \(x = y, x < y, x \in X, x \in P_a\)
- And \(\land, \lor, \neg, \exists, \forall\)

(FOL (\(<, \Sigma\)) is the restriction without set variables.)

The monadic second order logic MPL is the restriction of MSOL where the interpretation of set variables \(X\) ranges only over paths.

Monadic Second Order Logic

Consider the monadic second order logic MSOL (\(<, \Sigma\)) to express properties of \(\Sigma\)-labeled trees. It contains:
- individual variables \(x, y, z, \ldots\) (for the nodes)
- set variables \(X, Y, Z, \ldots\) (for set of nodes)
- predicate constants \(P_a\) for \(a \in \Sigma\)
- And \(x = y, x < y, x \in X, x \in P_a\)
- And \(\land, \lor, \neg, \exists, \forall\)

(FOL (\(<, \Sigma\)) is the restriction without set variables.)

The monadic path logic MPL is the restriction of MSOL where the interpretation of set variables \(X\) ranges only over paths.

Example of MSOL formulas

\(P_1\): characterizing the set of even states in \(T\)

We have to specify that:
- the root belongs to \(X\) \((\exists x \in X. \forall y. x < y \lor x = y)\)
- If \(y\) is a succ. of \(x\) in \(X\) \((x < y \land \forall u. \neg (x < u < y))\):
  - \(y\) is not in \(X\)
  - any successor of \(y\) is in \(X\)

Thus \(P_1\) can be written:

\[
\exists x. (\exists y. x < y \lor y = y) \land (\forall y. x < y \land \forall u. \neg(x < u < y) \Rightarrow (y \notin X \land \forall z. (y < z \land \forall u. \neg(y < u < z) \Rightarrow z \in X)))
\]

From CTL* to MPL

**Theorem**

For any \(\varphi \in CTL^*\), there exists \(F_\varphi \in MPL\) s.t. \(\varphi \equiv F_\varphi\)

\(F_\varphi\) is defined by induction.

A formula \(F_\varphi(x)\) is associated with every state formula \(\varphi\).

A formula \(F_\varphi(X)\) is associated with every path formula \(\varphi_p\).

For any tree \(T\) and any node \(s \in T\) and any path \(\rho\) in \(T\), we have:

\[
\begin{align*}
  s \models T \varphi & \iff (T, s) \models F_\varphi(x) \\
  \rho \models T \varphi_p & \iff (T, \rho) \models F_\varphi(X)
\end{align*}
\]

i.e. \(T \models F_\varphi(x \leftarrow s)\) and \(T \models F_\varphi(X \leftarrow \rho)\).
From **CTL** to **MPL**

**Theorem**

For any \( \varphi \in \text{CTL}^* \), there exists \( F_\varphi \in \text{MPL} \) s.t. \( \varphi \equiv F_\varphi \).

\( F_\varphi \) is defined by induction.

A formula \( F_\varphi(x) \) is associated with every state formula \( \varphi \).

A formula \( F_\varphi(p)(x) \) is associated with every path formula \( \varphi_p \).

For any tree \( T \) and any node \( s \in T \) and any path \( p \in T \), we have:

\[ s \models_T \varphi \iff (T, s) \models F_\varphi(x) \]
\[ p \models_T \varphi_p \iff (T, p) \models F_\varphi(p)(X) \]

i.e. \( T \models F_\varphi(x \leftarrow s) \) and \( T \models F_\varphi(X \leftarrow p) \).

**From **CTL** to **MPL**

**Definition of** \( F_\varphi \) (Hafer & Thomas, 1987):

- \( F_\varphi(x) \) def= \( x \in P_s \)
- \( F_{E_\varphi}(x) \) def= \( \exists Y . \left( \text{Y starts at } x^* \land F_{\varphi_p}(Y) \right) \)
  - with "Y starts at \( x^* \): \( x \in Y \land \forall y \in Y (x < y \lor y = x) \)
- \( F_{X_\varphi}(Z) \) def= \( \exists 2x \exists Y . \left( \text{Y \subset Z} \land \text{Y starts at } y^* \land y \text{ is a successor of } z^* \land F_{\varphi_\psi}(Y) \right) \)
  - with: "Y \subset Z": \( \forall u \in Y, y \in Z \)
  - and: "y is a successor of \( z^* \): \( z < y \land \forall u \neg(z < u < y) \)
- \( F_{\varphi_\psi} \cup U_{\varphi_\psi}(Z) \) def= \( \exists Y . \left( \text{Y \subset Z} \land F_{\varphi_\psi}(Y) \land (\forall Y' Y \subset Y^* \land Y' \subset Z \Rightarrow F_{\varphi_\psi}(Y')) \right) \)

**From **MPL** to **CTL**

**Theorem** (Hafer & Thomas, 1987): **CTL** is expressively equivalent to the MPL sentences over full binary trees.

But this is not true in general.

\( \varphi \) def= \( \exists x \exists y \left( \neg(x < y \lor y < x) \land x \in P_s \land y \in P_s \right) \)

\( \varphi \) expresses that there are two incomparable states satisfying a.
From \textit{MPL to CTL}^*

\textbf{Theorem (Moller & Rabinovich, 1999)}

\textit{CTL}^* is expressively equivalent to the \textit{MPL} sentences which respect bisimulation equivalence.

Main ideas of the proof...

- Let \textit{MPL}_n and \textit{FOL}_n be the restrictions to formulas with a quantifier depth less than \( n \).
- \( \equiv_n \) iff \( T \) and \( T' \) satisfy the same \textit{MPL}_n formulas.
- \( \equiv_n \) defines finitely many equivalence classes: \( C_1, \ldots, C_m \).
- Given a (in)finite path \( \rho \) in \( T, \nu_\rho(\rho) \) is a word over an extended alphabet \( \Sigma' \) that describes precisely \( \rho \) w.r.t. \textit{MPL}_n.
  
  \textit{Idea:} for every state along \( \rho \), we store its letter (in \( \Sigma \)) and the equivalence classes of all its subtrees.
  
  \( (\Sigma' = \Sigma \times P(\{1, \ldots, m\})) \).

From \textit{MPL} to \textit{CTL}^*

\textbf{Theorem (Moller & Rabinovich, 1999)}

\textit{CTL}^* is expressively equivalent to the \textit{MPL} sentences over full binary trees.

But this is not true in general!

\( \textit{MPL} \) respects the bisimulation equivalence but \textit{MPL} sentences do not:

\[ \Phi \overset{\text{def}}{=} \exists x \exists y \left( \neg(x < y \lor y < x) \land x \in P_a \land y \in P_a \right) \]

\( \Phi \) expresses that there are two uncomparable states satisfying \( a \).

---

From \textit{MPL} to \textit{CTL}^*

\textbf{Theorem (Moller & Rabinovich, 1999)}

\textit{CTL}^* is expressively equivalent to the \textit{MPL} sentences over full binary trees.

Main ideas of the proof...

- Let \textit{MPL}_n and \textit{FOL}_n be the restrictions to formulas with a quantifier depth less than \( n \).
- \( \equiv_n \) iff \( T \) and \( T' \) satisfy the same \textit{MPL}_n formulas.
- \( \equiv_n \) defines finitely many equivalence classes: \( C_1, \ldots, C_m \).
- Given a (in)finite path \( \rho \) in \( T, \nu_\rho(\rho) \) is a word over an extended alphabet \( \Sigma' \) that describes precisely \( \rho \) w.r.t. \textit{MPL}_n.
  
  \textit{Idea:} for every state along \( \rho \), we store its letter (in \( \Sigma \)) and the equivalence classes of all its subtrees.
  
  \( (\Sigma' = \Sigma \times P(\{1, \ldots, m\})) \).

---

From \textit{MPL} to \textit{CTL}^*

\textbf{Theorem (Moller & Rabinovich, 1999)}

\textit{CTL}^* is expressively equivalent to the \textit{MPL} sentences over full binary trees.

Main ideas of the proof...

- Let \textit{MPL}_n and \textit{FOL}_n be the restrictions to formulas with a quantifier depth less than \( n \).
- \( \equiv_n \) iff \( T \) and \( T' \) satisfy the same \textit{MPL}_n formulas.
- \( \equiv_n \) defines finitely many equivalence classes: \( C_1, \ldots, C_m \).
- Given a (in)finite path \( \rho \) in \( T, \nu_\rho(\rho) \) is a word over an extended alphabet \( \Sigma' \) that describes precisely \( \rho \) w.r.t. \textit{MPL}_n.
  
  \textit{Idea:} for every state along \( \rho \), we store its letter (in \( \Sigma \)) and the equivalence classes of all its subtrees.
  
  \( (\Sigma' = \Sigma \times P(\{1, \ldots, m\})) \).
From **MPL** to **CTL***(Moller & Rabinovich, 1999)**

A tree is **wide** if every subtree is reproduced an infinite number of times.
(Any tree \( T \) can be transformed into a wide tree \( T' \) and \( T \equiv T' \).)

The proof is based on a composition Theorem:

**Theorem (Moller & Rabinovich, 1999)**

For every MPL formula \( F(x) \) there is a FOL\((\langle, \Sigma \rangle)\) formula \( \Phi \), s.t. for any wide tree \( T \), and any node \( s \in T \), we have:

\[
(T, s) \models F(x) \iff \nu_\Sigma(\xi_{F \to s}) \models \Phi
\]

(\( \approx \) similar result for \( F(X) \) and \( \rho \in T \) . . .)

And use Kamp’s theorem to go from FOL to LTL: we can translate \( F(x) \) into \( \Phi F \in \text{CTL}^* \).

---

From **MPL to CTL***(Moller & Rabinovich, 1999)**

Given a MPL formula \( F \) invariant under bisimulation, then:

\[
T \models F
\]

\[
\Rightarrow \quad T' \models F \quad \quad \text{F inv. bisim.}
\]

\[
\Rightarrow \quad T' \models F' \quad \quad \Phi_F \in \text{CTL}^*, \text{cf previous slide}
\]

\[
\Rightarrow \quad T' \models F' \quad T \approx T' \text{ and } \text{CTL}^* \text{ resp. } \approx.
\]

Another result exists for the propositional \( \mu \)-calculus:

**Theorem (Janin & Walukiewicz, 1996)**

Propositional \( \mu \)-calculus is expressively equivalent to the MSOL sentences which respect bisimulation equivalence.

---

From **MPL to CTL***(Moller & Rabinovich, 1999)**

Given a MPL formula \( F \) invariant under bisimulation, then:

\[
T \models F
\]

\[
\Rightarrow \quad T' \models F \quad \quad \text{F inv. bisim.}
\]

\[
\Rightarrow \quad T' \models F' \quad \quad \Phi_F \in \text{CTL}^*, \text{cf previous slide}
\]

\[
\Rightarrow \quad T' \models F' \quad T \approx T' \text{ and } \text{CTL}^* \text{ resp. } \approx.
\]

Another result exists for the propositional \( \mu \)-calculus:

**Theorem (Janin & Walukiewicz, 1996)**

Propositional \( \mu \)-calculus is expressively equivalent to the MSOL sentences which respect bisimulation equivalence.

---

From **MPL to CTL***(Moller & Rabinovich, 1999)**

Given a MPL formula \( F \) invariant under bisimulation, then:

\[
T \models F
\]

\[
\Rightarrow \quad T' \models F \quad \quad \text{F inv. bisim.}
\]

\[
\Rightarrow \quad T' \models F' \quad \quad \Phi_F \in \text{CTL}^*, \text{cf previous slide}
\]

\[
\Rightarrow \quad T' \models F' \quad T \approx T' \text{ and } \text{CTL}^* \text{ resp. } \approx.
\]

Another result exists for the propositional \( \mu \)-calculus:

**Theorem (Janin & Walukiewicz, 1996)**

Propositional \( \mu \)-calculus is expressively equivalent to the MSOL sentences which respect bisimulation equivalence.
From MPL to CTL*(Moller & Rabinovich, 1999)

Given a MPL formula $F$ invariant under bisimulation, then:

- $A \models F$ if $F$ inv. sim.
- $A \models \psi_F$ for $\psi_F \in CTL^*$, cf previous slide
- $A \models \psi_F$ if $T \equiv T$ and $CTL^*$ resp. $\approx$.

Another result exists for the propositional $\mu$-calculus:

**Theorem (Janin & Walukiewicz, 1996)**

Propositional $\mu$-calculus is expressively equivalent to the MSOL sentences which respect bisimulation equivalence.

### Additional results

**Theorem (Moller & Rabinovich, 2003)**

Counting-CTL$^*$ is expressively equivalent to MPL.

New modalities $D_n^*$

$s \models D_n^\varphi$ iff “for at least $n$ different $s \rightarrow s'$, we have $s' \models \varphi$.”

Let $BTL_4$ be the temporal logic defined with the modalities $E\varphi$ with $\varphi$ a first-order future formula with $qd(\varphi) \leq k$.

**Theorem (Rabinovich & Schnoebelen, 2000)**

$ECTL^+$ and $BTL_2$ have the same expressive power.

### Automata theory and branching-time logics

For any $\varphi \in LTL$, there exists a Büchi automaton $A_\varphi$ that recognizes the models of $\varphi$.
And $|A_\varphi|$ is in $2^{O(|\varphi|)}$

For any $\varphi \in LTL$, there exists an alternating Büchi automaton $A^*_\varphi$ that recognizes $M(\varphi)$.
And $|A^*_\varphi|$ is in $O(|\varphi|)$

And for $\varphi \in CTL$?

One can build an Alternating Tree Automaton that recognizes $\mathcal{M}(\varphi)$.
References: (Vardi, 1995), (Kupferman, Vardi, Wolper, 2000), (Wilke, 1999).

### Automata theory and branching-time logics

For any $\varphi \in LTL$, there exists a Büchi automaton $A_\varphi$ that recognizes the models of $\varphi$.
And $|A_\varphi|$ is in $2^{O(|\varphi|)}$

For any $\varphi \in LTL$, there exists an alternating Büchi automaton $A^*_\varphi$ that recognizes $M(\varphi)$.
And $|A^*_\varphi|$ is in $O(|\varphi|)$

And for $\varphi \in CTL$?

One can build an Alternating Tree Automaton that recognizes $\mathcal{M}(\varphi)$.
References: (Vardi, 1995), (Kupferman, Vardi, Wolper, 2000), (Wilke, 1999).
Automata theory and branching-time logics

For any $\varphi \in \text{LTL}$, there exists a Büchi automaton $A_\varphi$ that recognizes the models of $\varphi$. And $|A_\varphi|$ is in $2^{|\varphi|}$.

For any $\varphi \in \text{LTL}$, there exists an alternating Büchi automaton $A^a_\varphi$ that recognizes $M(\varphi)$. And $|A^a_\varphi|$ is in $O(|\varphi|)$.

And for $\varphi \in \text{CTL}$?

One can build an Alternating Tree Automaton that recognizes $M(\varphi)$.

References: (Vardi, 1995), (Kupferman, Vardi, Wolper, 2000), (Wilke, 1999).

Non-deterministic tree automata

Let $D \subseteq \mathbb{N}$ be a finite set of arities.
We consider automata on $\Sigma$-labeled leafless $D$-trees.

$A = (\Sigma, D, s_0, \rho, F)$

- $S$ : a finite set of states, and $s_0 \in S$.
- $F \subseteq S$ : a Büchi acceptance condition.
- $\rho : S \times \Sigma \times D \rightarrow 2^S$ : a transition function s.t. $\rho(s, a, k)$ is a set of $k$-tuples $(s_1, \ldots, s_k)$.

Let $T = (T, l)$ be a $\Sigma$-labeled $D$-tree.

A run $r : T \rightarrow S$ of $A$ on $T$ is an $S$-labeled $D$-tree s.t.

- $r(\varepsilon) = s_0$
- For any $x \in T$, $\text{arity}(x) = k$, we have $\langle r(x), \ldots, r(k) \rangle \in \rho(r(x), l(x), k)$
- For any branch $x_1 x_2 \ldots$, there are infinitely many $i$ s.t. $r(x_i) \in F$

$T(A) : \text{set of trees accepted by } A.$

Example of NDTA

$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\})$ with

$\rho(s_0, a, 2) = \{s_1, s_1\}$, $\rho(s_0, b, 2) = \{s_0, s_2\}$,
$\rho(s_1, a, 2) = \{s_1, s_1\}$, $\rho(s_1, b, 2) = \{s_0, s_2\}$.

Automata theory and branching-time logics

For any $\varphi \in \text{LTL}$, there exists a Büchi automaton $A_\varphi$ that recognizes the models of $\varphi$. And $|A_\varphi|$ is in $2^{|\varphi|}$.

For any $\varphi \in \text{LTL}$, there exists an alternating Büchi automaton $A^a_\varphi$ that recognizes $M(\varphi)$. And $|A^a_\varphi|$ is in $O(|\varphi|)$.

And for $\varphi \in \text{CTL}$?

One can build an Alternating Tree Automaton that recognizes $M(\varphi)$.

References: (Vardi, 1995), (Kupferman, Vardi, Wolper, 2000), (Wilke, 1999).

Non-deterministic tree automata

Let $D \subseteq \mathbb{N}$ be a finite set of arities.
We consider automata on $\Sigma$-labeled leafless $D$-trees.

$A = (\Sigma, D, s_0, \rho, F)$

- $S$ : a finite set of states, and $s_0 \in S$.
- $F \subseteq S$ : a Büchi acceptance condition.
- $\rho : S \times \Sigma \times D \rightarrow 2^S$ : a transition function s.t. $\rho(s, a, k)$ is a set of $k$-tuples $(s_1, \ldots, s_k)$.

Let $T = (T, l)$ be a $\Sigma$-labeled $D$-tree.

A run $r : T \rightarrow S$ of $A$ on $T$ is an $S$-labeled $D$-tree s.t.

- $r(\varepsilon) = s_0$
- For any $x \in T$, $\text{arity}(x) = k$, we have $\langle r(x), \ldots, r(k) \rangle \in \rho(r(x), l(x), k)$
- For any branch $x_1 x_2 \ldots$, there are infinitely many $i$ s.t. $r(x_i) \in F$

$T(A) : \text{set of trees accepted by } A.$

Example of NDTA

$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\})$ with

$\rho(s_0, a, 2) = \{s_1, s_1\}$, $\rho(s_0, b, 2) = \{s_0, s_2\}$,
$\rho(s_1, a, 2) = \{s_1, s_1\}$, $\rho(s_1, b, 2) = \{s_0, s_2\}$.

Non-deterministic tree automata

Let $D \subseteq \mathbb{N}$ be a finite set of arities.
We consider automata on $\Sigma$-labeled leafless $D$-trees.

$A = (\Sigma, D, s_0, \rho, F)$

- $S$ : a finite set of states, and $s_0 \in S$.
- $F \subseteq S$ : a Büchi acceptance condition.
- $\rho : S \times \Sigma \times D \rightarrow 2^S$ : a transition function s.t. $\rho(s, a, k)$ is a set of $k$-tuples $(s_1, \ldots, s_k)$.

Let $T = (T, l)$ be a $\Sigma$-labeled $D$-tree.

A run $r : T \rightarrow S$ of $A$ on $T$ is an $S$-labeled $D$-tree s.t.

- $r(\varepsilon) = s_0$
- For any $x \in T$, $\text{arity}(x) = k$, we have $\langle r(x), \ldots, r(k) \rangle \in \rho(r(x), l(x), k)$
- For any branch $x_1 x_2 \ldots$, there are infinitely many $i$ s.t. $r(x_i) \in F$

$T(A) : \text{set of trees accepted by } A.$
Example of NDTA
\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\}) \]
with
\[ \rho(s_0, a, 2) = (s_1, s_1), \quad \rho(s_0, b, 2) = (s_0, s_0), \quad \rho(s_1, a, 2) = (s_1, s_1), \quad \rho(s_1, b, 2) = (s_0, s_0). \]

A recognizes infinite binary trees where any branch contains infinitely many \(a\).
Example of NDTA

\[ A = ((\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\}) \text{ with } \]
\[ \rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = ((s_1, s_0), (s_0, s_1)) \]
\[ \rho(s_1, b, 2) = \emptyset \]

A recognizes infinite binary trees where every node has an immediate successor labeled by \( a \).
Alternating Tree Automata

\[ \rho : S \times \Sigma \times D \rightarrow B^+(\mathbb{N} \times S) \]

with \( \rho(s, a, k) \in B^+({1, \ldots, k} \times S) \).

For ex. \( \rho(s, a, 3) = (1, s_1) \lor (2, s_1) \land (3, s_2) \)

NDTA: \( \rho'(s, a, k) = \bigvee_{(s_1, \ldots, s_k) \in \rho(s, a, k)} (1, s_1) \land (2, s_2) \land \ldots (k, s_k) \)

Example of ATA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\}) \]

with
\[ \rho(s_0, a, 2) = (1, s_1) \lor (2, s_1) \land (1, s_0) \land (2, s_0) \]
\[ \rho(s_1, a, 2) = (1, s_1) \land (2, s_1) \land (1, s_0) \land (2, s_0) \]
\[ \rho(s_0, b, 2) = (1, s_0) \land (2, s_0) \land (1, s_1) \land (2, s_1) \]
\[ \rho(s_1, b, 2) = (1, s_1) \land (2, s_1) \land (1, s_0) \land (2, s_0) \]
\[ \rho(s_0, a, 3) = (1, s_1) \lor (2, s_1) \land (3, s_2) \land (1, s_0) \land (2, s_0) \land (3, s_2) \]

A run on \( \Sigma \)-labeled leafless \( D \)-tree \( (T, l) \) is a \( (\mathbb{N}^* \times S) \)-labeled tree \( (T', l') \).

Each node of \( T \), corresponds to a node of \( T' \).

Label \( (x, s) \) : a copy of \( A \) reading the node \( x \) of \( T \) in state \( s \).

- \( l(c) = (c, s_0) \)
- If \( y \in T', l(y) = (x, s) \), arity \( x \) = \( k \) and \( \rho(s, l(x), k) = \theta \),
  then:
  \( \exists Q = \{(c_1, s_1), \ldots, (c_n, s_n)\} \subseteq \{1, \ldots, k\} \times S \) s.t.
  \( Q \models \theta \) and \( \forall 1 \leq i \leq n, we have:
  \( y \cdot i \in T' \) and \( l(y \cdot i) = (x \cdot c_i, s_i) \)
Example of ATA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\}) \]

\[ \rho(s_0, a, 2) = ((1 \land s_1) \lor (2, s_1)) \land (1, s_0) \land (2, s_0) \]

\[ \rho(s_1, b, 2) = \top \]

Alternating Trees Automata for CTL (Vardi, 1995)

Let \( \varphi \) be a CTL formula in positive normal form, and \( D \subseteq \mathbb{N} \).

\[ A_{D, \varphi} = (2^{AP}, D, \text{SubF}(\varphi), \varphi, F) \]

- \( \rho(P, a, k) = \bot \) if \( P \in a \)
- \( \rho(P, a, k) = \top \) if \( P \notin a, \ldots \)
- \( \rho(\varphi_1 \land \varphi_2, a, k) = \rho(\varphi_1, a, k) \land \rho(\varphi_2, a, k) \)
- \( \rho(\text{EX} \varphi_1, a, k) = \bigvee_{c=1}^{\ldots k}(c, \varphi_1) \)
- \( \rho(\text{AX} \varphi_1, a, k) = \bigwedge_{c=1}^{\ldots k}(c, \varphi_1) \)
- \( \rho(\text{E}^{c_1} \text{U} \varphi_2, a, k) = \rho(\varphi_1, a, k) \lor (\rho(\varphi_1, a, k) \land \bigvee_{c=1}^{\ldots k}(c, \text{E}^{c_1} \text{U} \varphi_2)) \)

And \( F \) is the set of \( \mathbb{W} \)-formulae in \( \varphi \).

Theorem (Kupferman, Vardi & Wolper, 2000)

\[ T(A_{D, \varphi}) \] is the set of \( D \)-trees satisfying \( \varphi \).
Decision procedures for CTL

Theorem (Emerson & Sistla, 1984)
A CTL formula $\phi$ is satisfiable iff it is satisfied in an $\{n\}$-tree where $n$ is the number of $E$ in $\phi$.

Satisfiability checking $\rightarrow$ non-emptiness checking of $A_{\{n\},\phi}$.
(in exponential time)

Model checking: $S \models \phi$?
- construct $A_{D_{\{n\},\phi}}$ (weak alternating tree automaton)
- construct $A_{S,\phi} = A_{D_{\{n\},\phi}} \times S$ (one-letter weak alternating word automaton)
- emptiness checking of $A_{S,\phi}$ (linear time !)

These algorithms are optimal.
Other constructions are possible for $CTL^*$ and the $\mu$-calculus.

Outline

1. BT-temporal logics with Past
2. $CTL^*$ vs Monadic second order logic
3. Automata theory and BT-temporal logics
4. Alternating-time temporal logic

Other results

There are many branching-time temporal logics.

$ATL$ (Alternating-time Temporal Logic) extends $CTL$ by considering strategies of agents.

Instead of quantifying over paths, we can quantify over the ability of some agents to ensure a given property.
(. . . whatever the choices of the other players.)

The same techniques can be applied.

The results may be quite different: it is important to consider carefully expressivity of TL.
A CGS $C$ is a 6-tuple $(Q, AP, l, Agt, Mov, \rightarrow)$ s.t:
- $Q$: a finite set of locations;
- $AP$: atomic propositions;
- $l$: $Q \rightarrow 2^{AP}$: a labeling function;
- $Agt = \{A_1, \ldots, A_k\}$: a set of agents (or players);
- $Mov: Q \times Agt \rightarrow \mathbb{N}_{\geq 1}$ the choice function. $Mov(\ell, A_i) =$ number of possible moves for $A_i$ from $\ell$.
- $\rightarrow: Q \times \mathbb{N}^k \rightarrow Q$: the transition table.

From a location $\ell$, each $A_i$ chooses some $m_{A_i}$ with $m_{A_i} < Mov(\ell, A_i)$.
$\rightarrow(\ell, m_{A_1}, \ldots, m_{A_k})$ gives the new location.

Notations:
- $Next(\ell) = \{\rightarrow(\ell, \ldots, m_{A_i}, \ldots) \mid \forall m_{A_i} \cdot 1 \leq i \leq k\}$
- $Next(\ell, A_j, m) = \{\rightarrow(\ell, \ldots, m_{A_{j-1}}$, $m$, $m_{A_{j+1}}$, ...)\}
Strategy and outcomes

Definition
- A **computation** is an infinite sequence $\rho = \ell_0 \ell_1 \cdots$ such that $\forall i, \ell_{i+1} \in \text{Next}\{\ell_i\}$.
- A **strategy** is a function $f_A$ s.t.
  $f_A(\ell_0, \cdots, \ell_m)$ is a possible move for $A_i$ from $\ell_m$.
- The **outcomes** $\text{Out}(\ell, f_A)$ are the set of computations from $\ell$ induced by the strategy $f_A$ for $A_i$.
- Given $A \subseteq \text{Agt}$ we note:
  $F_A = \{f_A(\ell_i) \mid \ell_i \in A_i\}$
  $\text{Out}(\ell, F_A)$

Syntax of ATL

Definition (Alur, Henzinger & Kupferman, 1997)
The syntax of **ATL** is defined by the following grammar:
$$\text{ATL} \ni \varphi_s, \psi_s ::= P \mid \neg \varphi_s \mid \varphi_s \lor \psi_s \mid \langle\langle A\rangle\rangle \varphi_p$$
with $P \in \text{AP}$ and $A \subseteq \text{Agt}$.

- $E = \langle\langle \text{Agt}\rangle\rangle$
- $A = \langle\langle \emptyset\rangle\rangle$

Semantics

Definition
- $\ell \models \langle\langle A\rangle\rangle \varphi_p$ iff $\exists F_A \in \text{Strat}(A) \forall \rho \in \text{Out}(\ell, F_A) \rho \models \varphi_p$
- $\rho \models \varphi_s U \psi_s$ iff $\exists i. \rho[i] \models \psi_s$ and $\forall 0 \leq j < i. \rho[j] \models \varphi_s$

- Abbreviation: $\Box[A] \varphi$ for $\neg \langle\langle A\rangle\rangle \neg \varphi$
- $\neg \langle\langle A\rangle\rangle \varphi \Rightarrow \langle\langle \text{Agt} \setminus A\rangle\rangle \neg \varphi$

Until vs. Weak Until

Definition
- $\varphi \psi \equiv \varphi \lor \psi \lor G \varphi$
- $\neg (\varphi \psi) \equiv (\neg \psi) \psi \lor (\neg \varphi \land \neg \psi)$

Theorem
$E\varphi \psi \equiv E \psi \lor E\varphi \psi$

Theorem (Laroussinie, Markey & Oreiby, 2006)
$\langle\langle A\rangle\rangle (a \psi b)$ cannot be expressed in ATL.
$\langle\langle A\rangle\rangle (a \psi b) \iff \langle\langle A\rangle\rangle a \lor \langle\langle A\rangle\rangle a U b$
Proof

$s' \models \langle A \rangle a W b$ but $s_i \not\models \langle A \rangle a W b$

Lemma

$\forall i > 0, \forall \psi \in ATL with |\psi| \leq i we have: s_i \models \psi$ iff $s'_i \models \psi$. 