

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

Expressiveness of Temporal Logics

François Laroussinie and Nicolas Markey

Lab. Specification et Verification
ENS Cachan & CNRS, France

August 3, 2006

⏪ ⏩ ⏴ ⏵

⏪ ⏩ ⏴ ⏵

CTL^* + Past

Definition

$$\begin{aligned} PCTL^* \ni \varphi, \psi & ::= P_1 \mid \dots \mid \neg \varphi \mid \varphi \wedge \psi \mid \mathbf{E}\varphi_P \mid \mathbf{A}\varphi_P \\ & \mid \varphi \mathbf{S} \psi \mid \mathbf{X}^{-1}\varphi \\ PCTL^*_p \ni \varphi_p, \psi_p & ::= \varphi \mid \neg \varphi_p \mid \varphi_p \wedge \psi_p \mid \mathbf{X}\varphi_p \mid \varphi_p \mathbf{U} \psi_p \end{aligned}$$

with $P \in AP$

$CTL + \mathbf{S}, \mathbf{X}^{-1}, CTL + \mathbf{F}^{-1}, ECTL + \mathbf{S}, \dots$

$PCTL^*$ 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.

⏪ ⏩ ⏴ ⏵

⏪ ⏩ ⏴ ⏵

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.

Semantics

$$\sigma \stackrel{\text{def}}{=} q_1 \cdots q_n$$

Definition

$\sigma \models_S P$	iff	$P \in I(q_n)$
$\sigma \models_S \varphi \wedge \psi$	iff	$\sigma \models_S \varphi$ and $\sigma \models_S \psi$
$\sigma \models_S \neg \varphi$	iff	$\sigma \not\models_S \varphi$
$\sigma \models_S \mathbf{E}\varphi_p$	iff	$\exists \rho \in \text{Exec}(q_n)$ s.t. $\sigma \cdot \rho, n \models_S \varphi_p$
$\sigma \models_S \mathbf{A}\varphi_p$	iff	$\forall \rho \in \text{Exec}(q_n)$ we have $\sigma \cdot \rho, n \models_S \varphi_p$
$\rho, n \models_S \varphi_p \mathbf{U} \psi_p$	iff	$\exists i \geq n, \rho, i \models_S \psi_p$ and $\forall n \leq j < i$, we have $\rho, j \models_S \varphi_p$
$\rho, n \models_S \mathbf{X} \varphi_p$	iff	$\rho, n+1 \models_S \varphi_p$
$\sigma \models_S \varphi \mathbf{S} \psi$	iff	$\exists 1 \leq i \leq n$, s.t. $\sigma _i \models \psi$ and $\forall i < j \leq n$, we have $\sigma _j \models \varphi$
$\sigma \models_S \mathbf{X}^{-1} \varphi$	iff	$n > 1$ and $\sigma _{n-1} \models \varphi$

Adding \mathbf{S} or \mathbf{X}^{-1} (Laroussinie & Schnoebelen, 1994)

L' is **initially** as expressive as L : $\forall \varphi \in L, \exists \varphi' \in L'$, such that for any **state** q in any KS, we have $q \models \varphi$ iff $q \models \varphi'$.

- $ECTL^+$ is not as expressive as $UB + \mathbf{S}$.
 $\mathbf{E}(a \vee b \mathbf{U} c) \mathbf{U} d$ can be expressed in $UB + \mathbf{S}$
- $ECTL^+$ is not as expressive as $UB + \mathbf{X}^{-1}$.
 $\mathbf{E}\mathbf{G}(a \vee \mathbf{X} a) \equiv_i \mathbf{E}\mathbf{G}(a \vee \mathbf{X}^{-1} a \rightarrow \mathbf{X}^{-1} \#)$

PAST may add expressivity !

Adding \mathbf{S} or \mathbf{X}^{-1} (Laroussinie & Schnoebelen, 1994)

L' is **initially** as expressive as L : $\forall \varphi \in L, \exists \varphi' \in L'$, such that for any **state** q in any KS, we have $q \models \varphi$ iff $q \models \varphi'$.

- $ECTL^+$ is not as expressive as $UB + \mathbf{S}$.
 $\mathbf{E}(a \vee b \mathbf{U} c) \mathbf{U} d$ can be expressed in $UB + \mathbf{S}$
- $ECTL^+$ is not as expressive as $UB + \mathbf{X}^{-1}$.
 $\mathbf{E}\mathbf{G}(a \vee \mathbf{X} a) \equiv_i \mathbf{E}\mathbf{G}(a \vee \mathbf{X}^{-1} a \rightarrow \mathbf{X}^{-1} \#)$

PAST may add expressivity !

Adding \mathbf{S} or \mathbf{X}^{-1} (Laroussinie & Schnoebelen, 1994)

L' is **initially** as expressive as L : $\forall \varphi \in L, \exists \varphi' \in L'$, such that for any **state** q in any KS, we have $q \models \varphi$ iff $q \models \varphi'$.

- $ECTL^+$ is not as expressive as $UB + \mathbf{S}$.
 $\mathbf{E}(a \vee b \mathbf{U} c) \mathbf{U} d$ can be expressed in $UB + \mathbf{S}$
- $ECTL^+$ is not as expressive as $UB + \mathbf{X}^{-1}$.
 $\mathbf{E}\mathbf{G}(a \vee \mathbf{X} a) \equiv_i \mathbf{E}\mathbf{G}(a \vee \mathbf{X}^{-1} a \rightarrow \mathbf{X}^{-1} \#)$

PAST may add expressivity !

Adding \mathbf{S} or \mathbf{X}^{-1} (Laroussinie & Schnoebelen, 1994)

L' is **initially** as expressive as L : $\forall \varphi \in L, \exists \varphi' \in L'$, such that for any **state** q in any KS, we have $q \models \varphi$ iff $q \models \varphi'$.

- $ECTL^+$ is not as expressive as $UB + \mathbf{S}$.
 $\mathbf{E}(a \vee b \mathbf{U} c) \mathbf{U} d$ can be expressed in $UB + \mathbf{S}$
- $ECTL^+$ is not as expressive as $UB + \mathbf{X}^{-1}$.
 $\mathbf{E}\mathbf{G}(a \vee \mathbf{X} a) \equiv_i \mathbf{E}\mathbf{G}(a \vee \mathbf{X}^{-1} a \rightarrow \mathbf{X}^{-1} \#)$

PAST may add expressivity !

Adding F^{-1}

$CTL + F^{-1}$ can be **weakly** separated.

$F^{-1}EF P$ 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 $CTL + F^{-1}$ formula can be separated.
Any $B(X, X^{-1}, S)$ formula can be separated.

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



Adding F^{-1}

$CTL + F^{-1}$ can be **weakly** separated.

$F^{-1}EF P$ 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 $CTL + F^{-1}$ formula can be separated.
Any $B(X, X^{-1}, S)$ formula can be separated.

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



Example of separation

$$\begin{aligned} E(P_1 \wedge F^{-1}P_2) U (P_3 \wedge F^{-1}P_4) &\equiv \\ (P_3 \wedge F^{-1}P_4) \vee & \\ (F^{-1}P_2 \wedge \dots & \\ F^{-1}P_4 \wedge EP_1 U P_3 \vee & \\ EP_1 U (P_1 \wedge P_4 \wedge EP_1 U P_3)) & \end{aligned}$$



Example of separation

$$\begin{aligned} E(P_1 \wedge F^{-1}P_2) U (P_3 \wedge F^{-1}P_4) &\equiv \\ (P_3 \wedge F^{-1}P_4) \vee & \\ (F^{-1}P_2 \wedge \dots & \\ F^{-1}P_4 \wedge EP_1 U P_3 \vee & \\ EP_1 U (P_1 \wedge P_4 \wedge EP_1 U P_3)) & \end{aligned}$$



Example of separation

$$\begin{aligned} E(P_1 \wedge F^{-1}P_2) U (P_3 \wedge F^{-1}P_4) &\equiv \\ (P_3 \wedge F^{-1}P_4) \vee & \\ (F^{-1}P_2 \wedge \dots & \\ F^{-1}P_4 \wedge EP_1 U P_3 \vee & \\ EP_1 U (P_1 \wedge P_4 \wedge EP_1 U P_3)) & \end{aligned}$$



Example of separation

$$\begin{aligned} E(P_1 \wedge F^{-1}P_2) U (P_3 \wedge F^{-1}P_4) &\equiv \\ (P_3 \wedge F^{-1}P_4) \vee & \\ (F^{-1}P_2 \wedge \dots & \\ F^{-1}P_4 \wedge EP_1 U P_3 \vee & \\ EP_1 U (P_1 \wedge P_4 \wedge EP_1 U P_3)) & \end{aligned}$$



Separation and initial equivalence

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

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

We have :

- $\varphi \mathbf{S} \psi \equiv_i \psi$ (1)
- $\mathbf{X}^{-1} \psi \equiv_i \perp$ (2)

By applying rules (1) and (2), we can easily deduce that Φ 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 \mathbf{F}^{-1} (Laroussinie & Schnoebelen, 1994)

The following results hold for **initial** equivalence.

- $\mathcal{B}(\mathbf{X})$ is as expressive as $\mathcal{B}(\mathbf{X}, \mathbf{X}^{-1}, \mathbf{S})$.
- CTL is as expressive as $CTL^+ + \mathbf{F}^{-1}$.
(but $CTL^+ + \mathbf{F}^{-1}$ is exponentially more succinct)
- $ECTL^+$ is as expressive as $ECTL^+ + \mathbf{F}^{-1}$.
- $ECTL + \mathbf{F}^{-1}$ is strictly more expressive than $ECTL$.
($\mathbf{EF}(a \wedge \mathbf{G}^{-1} b)$ cannot be expressed in $ECTL$)
- $ECTL + \mathbf{F}^{-1}$ is strictly less expressive than $ECTL^+$.
($\mathbf{E}(\tilde{\mathbf{F}} a \wedge \tilde{\mathbf{F}} b)$ cannot be expressed in $ECTL + \mathbf{F}^{-1}$)

BTL with \mathbf{F}^{-1} (Laroussinie & Schnoebelen, 1994)

The following results hold for **initial** equivalence.

- $\mathcal{B}(\mathbf{X})$ is as expressive as $\mathcal{B}(\mathbf{X}, \mathbf{X}^{-1}, \mathbf{S})$.
- CTL is as expressive as $CTL^+ + \mathbf{F}^{-1}$.
(but $CTL^+ + \mathbf{F}^{-1}$ is exponentially more succinct)
- $ECTL^+$ is as expressive as $ECTL^+ + \mathbf{F}^{-1}$.
- $ECTL + \mathbf{F}^{-1}$ is strictly more expressive than $ECTL$.
($\mathbf{EF}(a \wedge \mathbf{G}^{-1} b)$ cannot be expressed in $ECTL$)
- $ECTL + \mathbf{F}^{-1}$ is strictly less expressive than $ECTL^+$.
($\mathbf{E}(\tilde{\mathbf{F}} a \wedge \tilde{\mathbf{F}} b)$ cannot be expressed in $ECTL + \mathbf{F}^{-1}$)

BTL with \mathbf{F}^{-1} (Laroussinie & Schnoebelen, 1994)

The following results hold for **initial** equivalence.

- $\mathcal{B}(\mathbf{X})$ is as expressive as $\mathcal{B}(\mathbf{X}, \mathbf{X}^{-1}, \mathbf{S})$.
- CTL is as expressive as $CTL^+ + \mathbf{F}^{-1}$.
(but $CTL^+ + \mathbf{F}^{-1}$ is exponentially more succinct)
- $ECTL^+$ is as expressive as $ECTL^+ + \mathbf{F}^{-1}$.
- $ECTL + \mathbf{F}^{-1}$ is strictly more expressive than $ECTL$.
($\mathbf{EF}(a \wedge \mathbf{G}^{-1} b)$ cannot be expressed in $ECTL$)
- $ECTL + \mathbf{F}^{-1}$ is strictly less expressive than $ECTL^+$.
($\mathbf{E}(\tilde{\mathbf{F}} a \wedge \tilde{\mathbf{F}} b)$ cannot be expressed in $ECTL + \mathbf{F}^{-1}$)

BTL with \mathbf{F}^{-1} (Laroussinie & Schnoebelen, 1994)

The following results hold for **initial** equivalence.

- $\mathcal{B}(\mathbf{X})$ is as expressive as $\mathcal{B}(\mathbf{X}, \mathbf{X}^{-1}, \mathbf{S})$.
- CTL is as expressive as $CTL^+ + \mathbf{F}^{-1}$.
(but $CTL^+ + \mathbf{F}^{-1}$ is exponentially more succinct)
- $ECTL^+$ is as expressive as $ECTL^+ + \mathbf{F}^{-1}$.
- $ECTL + \mathbf{F}^{-1}$ is strictly more expressive than $ECTL$.
($\mathbf{EF}(a \wedge \mathbf{G}^{-1} b)$ cannot be expressed in $ECTL$)
- $ECTL + \mathbf{F}^{-1}$ is strictly less expressive than $ECTL^+$.
($\mathbf{E}(\tilde{\mathbf{F}} a \wedge \tilde{\mathbf{F}} b)$ cannot be expressed in $ECTL + \mathbf{F}^{-1}$)

BTL with \mathbf{F}^{-1} (Laroussinie & Schnoebelen, 1994)

The following results hold for **initial** equivalence.

- $\mathcal{B}(\mathbf{X})$ is as expressive as $\mathcal{B}(\mathbf{X}, \mathbf{X}^{-1}, \mathbf{S})$.
- CTL is as expressive as $CTL^+ + \mathbf{F}^{-1}$.
(but $CTL^+ + \mathbf{F}^{-1}$ is exponentially more succinct)
- $ECTL^+$ is as expressive as $ECTL^+ + \mathbf{F}^{-1}$.
- $ECTL + \mathbf{F}^{-1}$ is strictly more expressive than $ECTL$.
($\mathbf{EF}(a \wedge \mathbf{G}^{-1} b)$ cannot be expressed in $ECTL$)
- $ECTL + \mathbf{F}^{-1}$ is strictly less expressive than $ECTL^+$.
($\mathbf{E}(\tilde{\mathbf{F}} a \wedge \tilde{\mathbf{F}} b)$ cannot be expressed in $ECTL + \mathbf{F}^{-1}$)

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

◀ ▶ ⏪ ⏩

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.

◀ ▶ ⏪ ⏩

Monadic Second Order Logic

Consider the **monadic second order logic MSOL** (\langle, Σ) to express properties of Σ -labeled trees. It contains:

- individual variables x, y, z, \dots (for the nodes)
- set variables X, Y, Z, \dots (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 $\wedge, \vee, \neg, \exists, \forall$

(FOL (\langle, Σ) 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** (\langle, Σ) to express properties of Σ -labeled trees. It contains:

- individual variables x, y, z, \dots (for the nodes)
- set variables X, Y, Z, \dots (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 $\wedge, \vee, \neg, \exists, \forall$

(FOL (\langle, Σ) 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** (\langle, Σ) to express properties of Σ -labeled trees. It contains:

- individual variables x, y, z, \dots (for the nodes)
- set variables X, Y, Z, \dots (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 $\wedge, \vee, \neg, \exists, \forall$

(FOL (\langle, Σ) 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** (\langle, Σ) to express properties of Σ -labeled trees. It contains:

- individual variables x, y, z, \dots (for the nodes)
- set variables X, Y, Z, \dots (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 $\wedge, \vee, \neg, \exists, \forall$

(FOL (\langle, Σ) 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** (\langle, Σ) to express properties of Σ -labeled trees. It contains:

- individual variables x, y, z, \dots (for the nodes)
- set variables X, Y, Z, \dots (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 $\wedge, \vee, \neg, \exists, \forall$

(FOL (\langle, Σ) 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** (\langle, Σ) to express properties of Σ -labeled trees. It contains:

- individual variables x, y, z, \dots (for the nodes)
- set variables X, Y, Z, \dots (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 $\wedge, \vee, \neg, \exists, \forall$

(FOL (\langle, Σ) 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** (\langle, Σ) to express properties of Σ -labeled trees. It contains:

- individual variables x, y, z, \dots (for the nodes)
- set variables X, Y, Z, \dots (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 $\wedge, \vee, \neg, \exists, \forall$

(FOL (\langle, Σ) 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 ($\exists x \in X. \forall y. x < y \vee x = y$)
- If y is a succ. of $x \in X$ ($x < y \wedge \forall u : \neg(x < u < y)$):
 - y is not in X
 - any successor of y is in X

Thus P_1 can be written:

$$\exists X. \left(\exists x_0 \in X. \forall y. x_0 < y \vee x_0 = y \right) \wedge \left(\forall x \in X \forall y \left((x < y \wedge \forall u : \neg(x < u < y)) \Rightarrow (y \notin X \wedge \forall z (y < z \wedge \forall u : \neg(y < u < z)) \Rightarrow z \in X) \right) \right)$$

◀ ▶ ◀ ▶

Example of MSOL formulas

P_2 : characterizing an infinite path in T

We have to specify that:

- the root is in the set X .
- any two nodes in X are $<$, $>$ or $=$.
- any node in X has a successor in X .

Thus :

$$\exists X. \left(\exists x_0 \in X. \forall y : x_0 < y \vee x_0 = y \right) \wedge \left(\forall x, y \in X : (x < y \vee y < x \vee x = y) \right) \wedge \left(\forall x \in X : \exists y \in X. (x < y \wedge \forall u : \neg(x < u < y)) \right)$$

◀ ▶ ◀ ▶

From CTL* to MPL

Theorem

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

F_φ is defined by induction.

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

A formula $F_{\varphi_p}(X)$ is associated with every path formula φ_p .

For any tree T and any node $s \in T$ and any path ρ in T , we have:

$$s \models_T \varphi \Leftrightarrow (T, s) \models F_\varphi(x)$$

$$\rho \models_T \varphi_p \Leftrightarrow (T, \rho) \models F_{\varphi_p}(X)$$

i.e. $T \models F_\varphi(x \leftarrow s)$ and $T \models F_{\varphi_p}(X \leftarrow \rho)$.

◀ ▶ ◀ ▶

From CTL* to MPL

Theorem

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

F_φ is defined by induction.

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

A formula $F_{\varphi_p}(X)$ is associated with every **path** formula φ_p .

For any tree T and any node $s \in T$ and any path ρ in T , we have:

$$\begin{aligned} s \models_T \varphi &\Leftrightarrow (T, s) \models F_\varphi(x) \\ \rho \models_T \varphi_p &\Leftrightarrow (T, \rho) \models F_{\varphi_p}(X) \end{aligned}$$

i.e. $T \models F_\varphi(x \leftarrow s)$ and $T \models F_{\varphi_p}(X \leftarrow \rho)$.

⋮

From CTL* to MPL

Definition of F_φ (Hafer & Thomas, 1987):

- $F_a(x) \stackrel{\text{def}}{=} x \in P_a$
- $F_{E\varphi_p}(x) \stackrel{\text{def}}{=} \exists Y. ("Y \text{ starts at } x" \wedge F_{\varphi_p}(Y))$
with "Y starts at x" $\stackrel{\text{def}}{=} x \in Y \wedge \forall y \in Y (x < y \vee x = y)$
- $F_{X\varphi_p}(Z) \stackrel{\text{def}}{=} \exists z \exists y \exists Y. ("Y \subseteq Z" \wedge$
"Y starts at y" \wedge "y is a successor of z" $\wedge F_{\varphi_p}(Y))$
with: "Y \subseteq Z" $\stackrel{\text{def}}{=} \forall u \in Y. y \in Z$
and: "y is a successor of z" $\stackrel{\text{def}}{=} z < y \wedge \forall u. \neg(z < u < y)$
- $F_{\varphi_p \cup \psi_p}(Z) \stackrel{\text{def}}{=} \exists Y. ("Y \subseteq Z" \wedge F_{\psi_p}(Y) \wedge (\forall Y' Y \subseteq Y' \wedge Y' \subseteq Z \Rightarrow F_{\varphi_p}(Y')))$

⋮

From CTL* to MPL

Definition of F_φ (Hafer & Thomas, 1987):

- $F_a(x) \stackrel{\text{def}}{=} x \in P_a$
- $F_{E\varphi_p}(x) \stackrel{\text{def}}{=} \exists Y. ("Y \text{ starts at } x" \wedge F_{\varphi_p}(Y))$
with "Y starts at x" $\stackrel{\text{def}}{=} x \in Y \wedge \forall y \in Y (x < y \vee x = y)$
- $F_{X\varphi_p}(Z) \stackrel{\text{def}}{=} \exists z \exists y \exists Y. ("Y \subseteq Z" \wedge$
"Y starts at y" \wedge "y is a successor of z" $\wedge F_{\varphi_p}(Y))$
with: "Y \subseteq Z" $\stackrel{\text{def}}{=} \forall u \in Y. y \in Z$
and: "y is a successor of z" $\stackrel{\text{def}}{=} z < y \wedge \forall u. \neg(z < u < y)$
- $F_{\varphi_p \cup \psi_p}(Z) \stackrel{\text{def}}{=} \exists Y. ("Y \subseteq Z" \wedge F_{\psi_p}(Y) \wedge (\forall Y' Y \subseteq Y' \wedge Y' \subseteq Z \Rightarrow F_{\varphi_p}(Y')))$

⋮

From CTL* to MPL

Definition of F_φ (Hafer & Thomas, 1987):

- $F_a(x) \stackrel{\text{def}}{=} x \in P_a$
- $F_{E\varphi_p}(x) \stackrel{\text{def}}{=} \exists Y. ("Y \text{ starts at } x" \wedge F_{\varphi_p}(Y))$
with "Y starts at x" $\stackrel{\text{def}}{=} x \in Y \wedge \forall y \in Y (x < y \vee x = y)$
- $F_{X\varphi_p}(Z) \stackrel{\text{def}}{=} \exists z \exists y \exists Y. ("Y \subseteq Z" \wedge$
"Y starts at y" \wedge "y is a successor of z" $\wedge F_{\varphi_p}(Y))$
with: "Y \subseteq Z" $\stackrel{\text{def}}{=} \forall u \in Y. y \in Z$
and: "y is a successor of z" $\stackrel{\text{def}}{=} z < y \wedge \forall u. \neg(z < u < y)$
- $F_{\varphi_p \cup \psi_p}(Z) \stackrel{\text{def}}{=} \exists Y. ("Y \subseteq Z" \wedge F_{\psi_p}(Y) \wedge (\forall Y' Y \subseteq Y' \wedge Y' \subseteq Z \Rightarrow F_{\varphi_p}(Y')))$

⋮

From CTL* to MPL

Definition of F_φ (Hafer & Thomas, 1987):

- $F_a(x) \stackrel{\text{def}}{=} x \in P_a$
- $F_{E\varphi_p}(x) \stackrel{\text{def}}{=} \exists Y. ("Y \text{ starts at } x" \wedge F_{\varphi_p}(Y))$
with "Y starts at x" $\stackrel{\text{def}}{=} x \in Y \wedge \forall y \in Y (x < y \vee x = y)$
- $F_{X\varphi_p}(Z) \stackrel{\text{def}}{=} \exists z \exists y \exists Y. ("Y \subseteq Z" \wedge$
"Y starts at y" \wedge "y is a successor of z" $\wedge F_{\varphi_p}(Y))$
with: "Y \subseteq Z" $\stackrel{\text{def}}{=} \forall u \in Y. y \in Z$
and: "y is a successor of z" $\stackrel{\text{def}}{=} z < y \wedge \forall u. \neg(z < u < y)$
- $F_{\varphi_p \cup \psi_p}(Z) \stackrel{\text{def}}{=} \exists Y. ("Y \subseteq Z" \wedge F_{\psi_p}(Y) \wedge (\forall Y' Y \subseteq Y' \wedge Y' \subseteq Z \Rightarrow F_{\varphi_p}(Y')))$

⋮

From MPL to CTL*

Theorem (Hafer & Thomas, 1987)

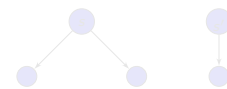
CTL is expressively equivalent to the MPL sentences over full binary trees.*

But this is not true in general !

CTL* respects the bisimulation equivalence but MPL sentences do not:

$$\Phi \stackrel{\text{def}}{=} \exists x \exists y (\neg(x < y \vee x = y \vee y < x) \wedge x \in P_a \wedge y \in P_a)$$

Φ expresses that there are two incomparable states satisfying a.



⋮

From MPL to CTL*

Theorem (Hafer & Thomas, 1987)

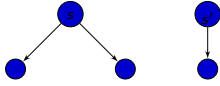
CTL* is expressively equivalent to the MPL sentences over **full binary trees**.

But this is not true in general !

CTL* respects the bisimulation equivalence but MPL sentences do not:

$$\Phi \stackrel{\text{def}}{=} \exists x \exists y \left(\neg(x < y \vee x = y \vee y < x) \wedge x \in P_a \wedge y \in P_a \right)$$

Φ expresses that there are two incomparable states satisfying a .



⏪ ⏩ ⏴ ⏵

From MPL to CTL*

Theorem (Moller & Rabinovich, 1999)

CTL* is expressively equivalent to the MPL sentences which respect bisimulation equivalence.

Main ideas of the proof...

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

⏪ ⏩ ⏴ ⏵

From MPL to CTL*

Theorem (Moller & Rabinovich, 1999)

CTL* is expressively equivalent to the MPL sentences which respect bisimulation equivalence.

Main ideas of the proof...

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

⏪ ⏩ ⏴ ⏵

From MPL to CTL*

Theorem (Moller & Rabinovich, 1999)

CTL* is expressively equivalent to the MPL sentences which respect bisimulation equivalence.

Main ideas of the proof...

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

⏪ ⏩ ⏴ ⏵

From MPL to CTL*

Theorem (Moller & Rabinovich, 1999)

CTL* is expressively equivalent to the MPL sentences which respect bisimulation equivalence.

Main ideas of the proof...

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

⏪ ⏩ ⏴ ⏵

From MPL to CTL*

Theorem (Moller & Rabinovich, 1999)

CTL* is expressively equivalent to the MPL sentences which respect bisimulation equivalence.

Main ideas of the proof...

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

⏪ ⏩ ⏴ ⏵

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 \bar{T} and $T \approx \bar{T}$.)

The proof is based on a composition Theorem:

Theorem (Moller & Rabinovich, 1999)

For every MPL_n formula $F(x)$ there is a $FOL_n(<, \Sigma')$ formula α , s.t. for any **wide** tree T , and any node $s \in T$, we have:

$$(T, s) \models F(x) \Leftrightarrow \nu_n(\varepsilon_T \rightarrow s) \models \alpha$$

(+ similar result for $F(X)$ and $\rho \in T \dots$)

And use Kamp's theorem to go from FOL to LTL: we can translate $F(x)$ into $\Phi_F \in CTL^*$.



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 \bar{T} and $T \approx \bar{T}$.)

The proof is based on a composition Theorem:

Theorem (Moller & Rabinovich, 1999)

For every MPL_n formula $F(x)$ there is a $FOL_n(<, \Sigma')$ formula α , s.t. for any **wide** tree T , and any node $s \in T$, we have:

$$(T, s) \models F(x) \Leftrightarrow \nu_n(\varepsilon_T \rightarrow s) \models \alpha$$

(+ similar result for $F(X)$ and $\rho \in T \dots$)

And use Kamp's theorem to go from FOL to LTL: we can translate $F(x)$ into $\Phi_F \in CTL^*$.



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

Given a MPL formula F invariant under bisimulation, then:

$$\begin{aligned} T &\models F \\ \Leftrightarrow \bar{T} &\models F && F \text{ inv. bisim.} \\ \Leftrightarrow \bar{T} &\models \Phi_F && \Phi_F \in CTL^*, \text{ cf previous slide} \\ \Leftrightarrow T &\models \Phi_F && T \approx \bar{T} \text{ and } CTL^* \text{ resp. } \approx. \end{aligned}$$

Another result exists for the propositional μ -calculus:

Theorem (Janin & Walukiewicz, 1996)

Propositional μ -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:

$$\begin{aligned} T &\models F \\ \Leftrightarrow \bar{T} &\models F && F \text{ inv. bisim.} \\ \Leftrightarrow \bar{T} &\models \Phi_F && \Phi_F \in CTL^*, \text{ cf previous slide} \\ \Leftrightarrow T &\models \Phi_F && T \approx \bar{T} \text{ and } CTL^* \text{ resp. } \approx. \end{aligned}$$

Another result exists for the propositional μ -calculus:

Theorem (Janin & Walukiewicz, 1996)

Propositional μ -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:

$$\begin{aligned} T &\models F \\ \Leftrightarrow \bar{T} &\models F && F \text{ inv. bisim.} \\ \Leftrightarrow \bar{T} &\models \Phi_F && \Phi_F \in CTL^*, \text{ cf previous slide} \\ \Leftrightarrow T &\models \Phi_F && T \approx \bar{T} \text{ and } CTL^* \text{ resp. } \approx. \end{aligned}$$

Another result exists for the propositional μ -calculus:

Theorem (Janin & Walukiewicz, 1996)

Propositional μ -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:

$$\begin{aligned} T &\models F \\ \Leftrightarrow \bar{T} &\models F && F \text{ inv. bisim.} \\ \Leftrightarrow \bar{T} &\models \Phi_F && \Phi_F \in CTL^*, \text{ cf previous slide} \\ \Leftrightarrow T &\models \Phi_F && T \approx \bar{T} \text{ and } CTL^* \text{ resp. } \approx. \end{aligned}$$

Another result exists for the propositional μ -calculus:

Theorem (Janin & Walukiewicz, 1996)

Propositional μ -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:

$$\begin{aligned} T &\models F \\ \Leftrightarrow \bar{T} &\models F && F \text{ inv. bisim.} \\ \Leftrightarrow \bar{T} &\models \Phi_F && \Phi_F \in CTL^*, \text{ cf previous slide} \\ \Leftrightarrow T &\models \Phi_F && T \approx \bar{T} \text{ and } CTL^* \text{ resp. } \approx. \end{aligned}$$

Another result exists for the propositional μ -calculus:

Theorem (Janin & Walukiewicz, 1996)

Propositional μ -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_k be the temporal logic defined with the modalities $E\varphi$ with φ a first-order future formula with $qd(\varphi) \leq k$.

Theorem (Rabinovich & Schnoebelen, 2000)

ECTL⁺ and BTL₂ have the same expressive power.

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

Theorem (Rabinovich & Schnoebelen, 2000)

ECTL⁺ and BTL₂ have the same expressive power.

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

Automata theory and branching-time logics

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

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

For any $\varphi \in LTL$, there exists an alternating Büchi automaton A_φ^a that recognizes $\mathcal{M}(\varphi)$.
And $|A_\varphi^a|$ 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_φ that recognizes the models of φ .

And $|A_\varphi|$ is in $2^{O(|\varphi|)}$

For any $\varphi \in LTL$, there exists an **alternating** Büchi automaton A_φ^a that recognizes $\mathcal{M}(\varphi)$.

And $|A_\varphi^a|$ 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_φ that recognizes the models of φ .

And $|A_\varphi|$ is in $2^{O(|\varphi|)}$

For any $\varphi \in LTL$, there exists an **alternating** Büchi automaton A_φ^a that recognizes $\mathcal{M}(\varphi)$.

And $|A_\varphi^a|$ 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).

⏪ ⏩ ⏴ ⏵

Non-deterministic tree automata

Let $\mathcal{D} \subset \mathbb{N}$ be a finite set of arities.

We consider automata on Σ -labeled leafless \mathcal{D} -trees.

$A = (\Sigma, \mathcal{D}, S, s_0, \rho, F)$

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

Let $\mathcal{T} = (T, l)$ be a Σ -labeled \mathcal{D} -tree.

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

- $r(\varepsilon) = s_0$
- For any x s.t. $\text{arity}(x) = k$, we have $(r(x \cdot 1), \dots, r(x \cdot k)) \in \rho(r(x), l(x), k)$
- for any branch $x_1 x_2 \dots$, there are infinitely many i s.t. $r(x_i) \in F$

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

⏪ ⏩ ⏴ ⏵

Non-deterministic tree automata

Let $\mathcal{D} \subset \mathbb{N}$ be a finite set of arities.

We consider automata on Σ -labeled leafless \mathcal{D} -trees.

$A = (\Sigma, \mathcal{D}, S, s_0, \rho, F)$

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

Let $\mathcal{T} = (T, l)$ be a Σ -labeled \mathcal{D} -tree.

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

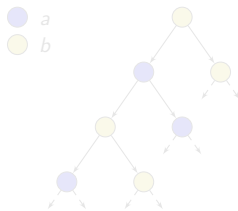
- $r(\varepsilon) = s_0$
- For any x s.t. $\text{arity}(x) = k$, we have $(r(x \cdot 1), \dots, r(x \cdot k)) \in \rho(r(x), l(x), k)$
- for any branch $x_1 x_2 \dots$, there are infinitely many i s.t. $r(x_i) \in F$

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

⏪ ⏩ ⏴ ⏵

Example of NDTA

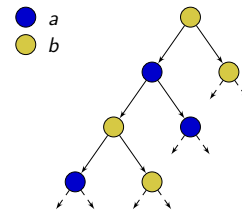
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\})$ with
 $\rho(s_0, a, 2) = (s_1, s_1)$, $\rho(s_0, b, 2) = (s_0, s_0)$,
 $\rho(s_1, a, 2) = (s_1, s_1)$, $\rho(s_1, b, 2) = (s_0, s_0)$.



⏪ ⏩ ⏴ ⏵

Example of NDTA

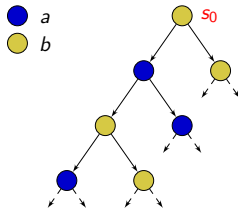
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\})$ with
 $\rho(s_0, a, 2) = (s_1, s_1)$, $\rho(s_0, b, 2) = (s_0, s_0)$,
 $\rho(s_1, a, 2) = (s_1, s_1)$, $\rho(s_1, b, 2) = (s_0, s_0)$.



⏪ ⏩ ⏴ ⏵

Example of NDTA

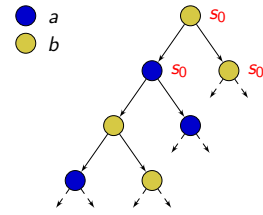
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\})$ with
 $\rho(s_0, a, 2) = (s_1, s_1)$, $\rho(s_0, b, 2) = (s_0, s_0)$,
 $\rho(s_1, a, 2) = (s_1, s_1)$, $\rho(s_1, b, 2) = (s_0, s_0)$.



⏪ ⏩ ⏴ ⏵

Example of NDTA

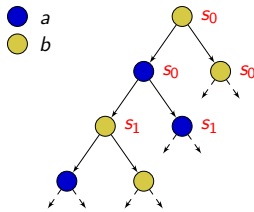
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\})$ with
 $\rho(s_0, a, 2) = (s_1, s_1)$, $\rho(s_0, b, 2) = (s_0, s_0)$,
 $\rho(s_1, a, 2) = (s_1, s_1)$, $\rho(s_1, b, 2) = (s_0, s_0)$.



⏪ ⏩ ⏴ ⏵

Example of NDTA

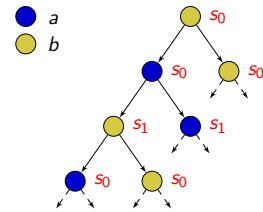
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\})$ with
 $\rho(s_0, a, 2) = (s_1, s_1)$, $\rho(s_0, b, 2) = (s_0, s_0)$,
 $\rho(s_1, a, 2) = (s_1, s_1)$, $\rho(s_1, b, 2) = (s_0, s_0)$.



⏪ ⏩ ⏴ ⏵

Example of NDTA

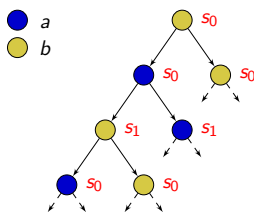
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\})$ with
 $\rho(s_0, a, 2) = (s_1, s_1)$, $\rho(s_0, b, 2) = (s_0, s_0)$,
 $\rho(s_1, a, 2) = (s_1, s_1)$, $\rho(s_1, b, 2) = (s_0, s_0)$.



⏪ ⏩ ⏴ ⏵

Example of NDTA

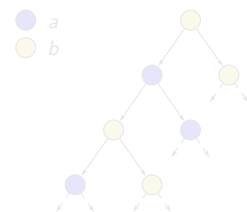
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\})$ with
 $\rho(s_0, a, 2) = (s_1, s_1)$, $\rho(s_0, b, 2) = (s_0, s_0)$,
 $\rho(s_1, a, 2) = (s_1, s_1)$, $\rho(s_1, b, 2) = (s_0, s_0)$.



⏪ ⏩ ⏴ ⏵

Example of NDTA

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

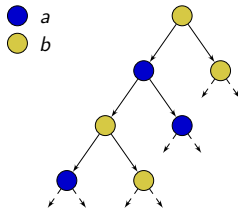


⏪ ⏩ ⏴ ⏵

A recognizes infinite binary trees where any branch contains infinitely many a.

Example of NDTA

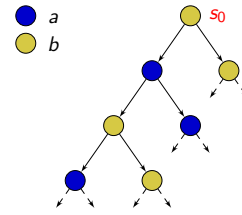
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\})$ with
 $\rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\}$
 $\rho(s_1, b, 2) = \emptyset$



← →

Example of NDTA

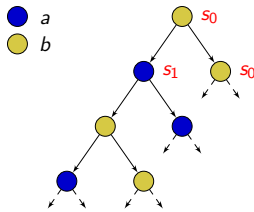
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\})$ with
 $\rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\}$
 $\rho(s_1, b, 2) = \emptyset$



← →

Example of NDTA

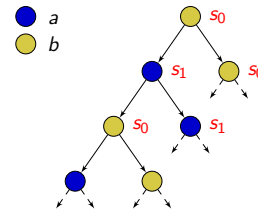
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\})$ with
 $\rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\}$
 $\rho(s_1, b, 2) = \emptyset$



← →

Example of NDTA

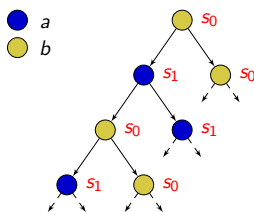
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\})$ with
 $\rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\}$
 $\rho(s_1, b, 2) = \emptyset$



← →

Example of NDTA

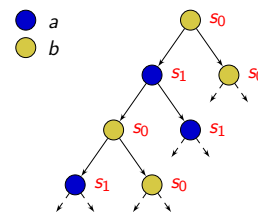
$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\})$ with
 $\rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\}$
 $\rho(s_1, b, 2) = \emptyset$



← →

Example of NDTA

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

$\rho : S \times \Sigma \times \mathcal{D} \rightarrow \mathcal{B}^+(\mathbb{N} \times S)$.
with $\rho(s, a, k) \in \mathcal{B}^+(\{1, \dots, k\} \times S)$.

For ex.: $\rho(s, a, 3) = ((1, s_1) \vee (2, s_1)) \wedge (3, s_2)$

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

Alternating Tree Automata

$\rho : S \times \Sigma \times \mathcal{D} \rightarrow \mathcal{B}^+(\mathbb{N} \times S)$.
with $\rho(s, a, k) \in \mathcal{B}^+(\{1, \dots, k\} \times S)$.

A run on Σ -labeled leafless \mathcal{D} -tree (T, l) is a $(\mathbb{N}^* \times S)$ -labeled tree (T_r, l_r) .

Each node of T_r corresponds to a node of T .

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

- $l_r(\varepsilon) = (\varepsilon, s_0)$
- If $y \in T_r$, $l_r(y) = (x, s)$, $\text{arity}(x) = k$ and $\rho(s, l(x), k) = \theta$, then:
There exists $Q = \{(c_1, s_1), \dots, (c_n, s_n)\} \subseteq \{1, \dots, k\} \times S$ s.t.
 $Q \models \theta$ and $\forall 1 \leq i \leq n$, we have:
 $y \cdot i \in T_r$ and $l_r(y \cdot i) = (x \cdot c_i, s_i)$

← →

← →

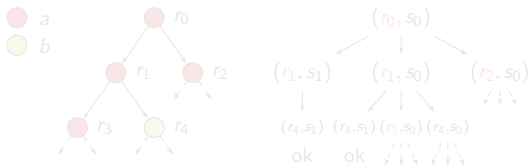
Example of ATA

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

$\rho(s_0, a, 2) = ((1, s_1) \vee (2, s_1)) \wedge (1, s_0) \wedge (2, s_0)$

$\rho(s_0, b, 2) = (1, s_0) \wedge (2, s_0)$

$\rho(s_1, a, 2) = (1, s_1) \vee (2, s_1)$ $\rho(s_1, b, 2) = \top$



← →

← →

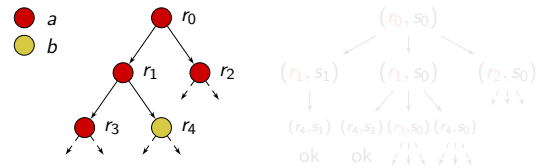
Example of ATA

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

$\rho(s_0, a, 2) = ((1, s_1) \vee (2, s_1)) \wedge (1, s_0) \wedge (2, s_0)$

$\rho(s_0, b, 2) = (1, s_0) \wedge (2, s_0)$

$\rho(s_1, a, 2) = (1, s_1) \vee (2, s_1)$ $\rho(s_1, b, 2) = \top$



← →

← →

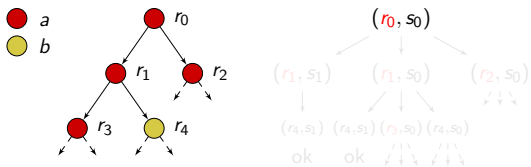
Example of ATA

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

$\rho(s_0, a, 2) = ((1, s_1) \vee (2, s_1)) \wedge (1, s_0) \wedge (2, s_0)$

$\rho(s_0, b, 2) = (1, s_0) \wedge (2, s_0)$

$\rho(s_1, a, 2) = (1, s_1) \vee (2, s_1)$ $\rho(s_1, b, 2) = \top$



← →

← →

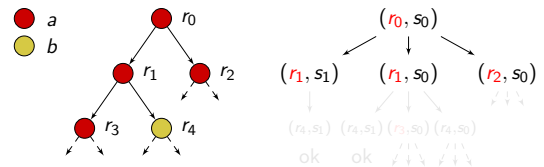
Example of ATA

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

$\rho(s_0, a, 2) = ((1, s_1) \vee (2, s_1)) \wedge (1, s_0) \wedge (2, s_0)$

$\rho(s_0, b, 2) = (1, s_0) \wedge (2, s_0)$

$\rho(s_1, a, 2) = (1, s_1) \vee (2, s_1)$ $\rho(s_1, b, 2) = \top$

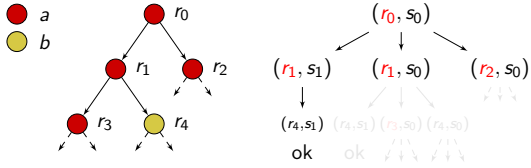


← →

← →

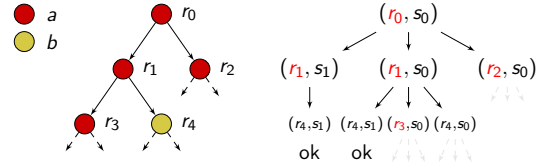
Example of ATA

$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\})$ with
 $\rho(s_0, a, 2) = ((1, s_1) \vee (2, s_1)) \wedge (1, s_0) \wedge (2, s_0)$
 $\rho(s_0, b, 2) = (1, s_0) \wedge (2, s_0)$
 $\rho(s_1, a, 2) = (1, s_1) \vee (2, s_1)$ $\rho(s_1, b, 2) = \top$



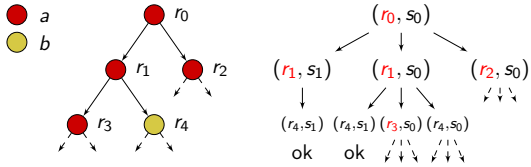
Example of ATA

$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\})$ with
 $\rho(s_0, a, 2) = ((1, s_1) \vee (2, s_1)) \wedge (1, s_0) \wedge (2, s_0)$
 $\rho(s_0, b, 2) = (1, s_0) \wedge (2, s_0)$
 $\rho(s_1, a, 2) = (1, s_1) \vee (2, s_1)$ $\rho(s_1, b, 2) = \top$



Example of ATA

$A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\})$ with
 $\rho(s_0, a, 2) = ((1, s_1) \vee (2, s_1)) \wedge (1, s_0) \wedge (2, s_0)$
 $\rho(s_0, b, 2) = (1, s_0) \wedge (2, s_0)$
 $\rho(s_1, a, 2) = (1, s_1) \vee (2, s_1)$ $\rho(s_1, b, 2) = \top$



Alternating Trees Automata for CTL (Vardi, 1995)

Let φ be a CTL formula in positive normal form, and $\mathcal{D} \subset \mathbb{N}$.

$A_{\mathcal{D}, \varphi} = (2^{AP}, \mathcal{D}, \text{SubF}(\varphi), \varphi, \rho, F)$:

- $\rho(P, a, k) = \#$ if $P \in a$, $\rho(P, a, k) = \text{ff}$ if $P \notin a$, ...
- $\rho(\varphi_1 \wedge \varphi_2, a, k) = \rho(\varphi_1, a, k) \wedge \rho(\varphi_2, a, k)$
- $\rho(\text{EX } \varphi_1, a, k) = \bigvee_{c=1, \dots, k} (c, \varphi_1)$
- $\rho(\text{AX } \varphi_1, a, k) = \bigwedge_{c=1, \dots, k} (c, \varphi_1)$
- $\rho(\text{E}\varphi_1 \text{ U } \varphi_2, a, k) = \rho(\varphi_2, a, k) \vee (\rho(\varphi_1, a, k) \wedge \bigvee_{c=1, \dots, k} (c, \text{E}\varphi_1 \text{ U } \varphi_2))$
- $\text{A}\varphi_1 \text{ U } \varphi_2, \text{E}\varphi_1 \text{ W } \varphi_2, \text{A}\varphi_1 \text{ W } \varphi_2 \dots$

And F is the set of **W**-formulae in φ .

Theorem (Kupferman, Vardi & Wolper, 2000)

$T(A_{\mathcal{D}, \varphi})$ is the set of \mathcal{D} -trees satisf. φ .

Alternating Trees Automata for CTL (Vardi, 1995)

Let φ be a CTL formula in positive normal form, and $\mathcal{D} \subset \mathbb{N}$.

$A_{\mathcal{D}, \varphi} = (2^{AP}, \mathcal{D}, \text{SubF}(\varphi), \varphi, \rho, F)$:

- $\rho(P, a, k) = \#$ if $P \in a$, $\rho(P, a, k) = \text{ff}$ if $P \notin a$, ...
- $\rho(\varphi_1 \wedge \varphi_2, a, k) = \rho(\varphi_1, a, k) \wedge \rho(\varphi_2, a, k)$
- $\rho(\text{EX } \varphi_1, a, k) = \bigvee_{c=1, \dots, k} (c, \varphi_1)$
- $\rho(\text{AX } \varphi_1, a, k) = \bigwedge_{c=1, \dots, k} (c, \varphi_1)$
- $\rho(\text{E}\varphi_1 \text{ U } \varphi_2, a, k) = \rho(\varphi_2, a, k) \vee (\rho(\varphi_1, a, k) \wedge \bigvee_{c=1, \dots, k} (c, \text{E}\varphi_1 \text{ U } \varphi_2))$
- $\text{A}\varphi_1 \text{ U } \varphi_2, \text{E}\varphi_1 \text{ W } \varphi_2, \text{A}\varphi_1 \text{ W } \varphi_2 \dots$

And F is the set of **W**-formulae in φ .

Theorem (Kupferman, Vardi & Wolper, 2000)

$T(A_{\mathcal{D}, \varphi})$ is the set of \mathcal{D} -trees satisf. φ .

Example

$\varphi = \text{AF AG } P$

$\varphi \equiv \text{AF} (\text{A } P \text{ W } \text{ff})$

$A_{\mathcal{D}, \varphi} = (\{\{P\}, \emptyset\}, \mathcal{D}, \{\varphi, \text{AG } P, P\}, \varphi, \rho, \{\text{AG } P\})$

S	$\rho(s, \{P\}, k)$	$\rho(s, \emptyset, k)$
P	$\#$	ff
$\text{AG } P$	$\bigwedge_{c=1, \dots, k} (c, \text{AG } P)$	ff
φ	$\bigwedge_{c=1, \dots, k} (c, \text{AG } P) \vee \bigwedge_{i=1, \dots, k} (c, \varphi)$	$\bigwedge_{i=1, \dots, k} (c, \varphi)$

Decision procedures for CTL

Theorem (Emerson & Sistla, 1984)

A CTL formula φ is satisfiable iff it is satisfied in an $\{n\}$ -tree where n is the number of **E** in φ .

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

Model checking: $S \models \varphi$?

- construct $A_{D_{S,\varphi}}$
(weak alternating tree automaton)
- construct $A_{S,\varphi} = A_{D_{S,\varphi}} \times S$
(one-letter weak alternating word automaton)
- emptiness checking of $A_{S,\varphi}$ (linear time !)

These algorithms are optimal.

Other constructions are possible for CTL* and the μ -calculus. 

Decision procedures for CTL

Theorem (Emerson & Sistla, 1984)


A CTL formula φ is satisfiable iff it is satisfied in an $\{n\}$ -tree where n is the number of **E** in φ .

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

Model checking: $S \models \varphi$?

- construct $A_{D_{S,\varphi}}$
(weak alternating tree automaton)
- construct $A_{S,\varphi} = A_{D_{S,\varphi}} \times S$
(one-letter weak alternating word automaton)
- emptiness checking of $A_{S,\varphi}$ (linear time !)

These algorithms are optimal.

Other constructions are possible for CTL* and the μ -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.



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.



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.



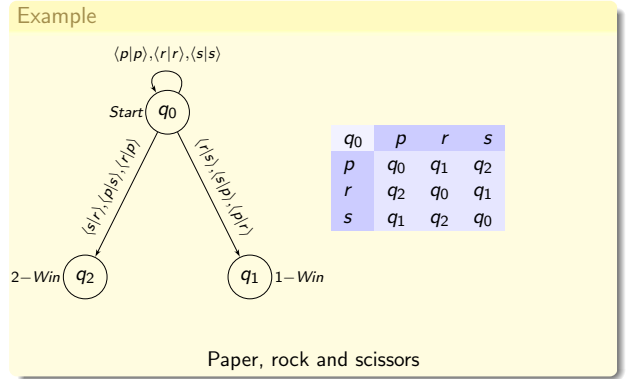
CGS definition

Definition

A CGS \mathcal{C} is a 6-tuple $(Q, AP, l, \text{Agt}, \text{Mov}, \rightarrow)$ s.t:

- Q : a finite set of locations;
- AP : atomic propositions;
- $l: Q \rightarrow 2^{AP}$: a labeling function;
- $\text{Agt} = \{A_1, \dots, A_k\}$: a set of agents (or players);
- $\text{Mov}: Q \times \text{Agt} \rightarrow \mathbb{N}_{\geq 1}$ the choice function. $\text{Mov}(\ell, A_i) =$ number of possible moves for A_i from ℓ .
- $\rightarrow: Q \times \mathbb{N}^k \rightarrow Q$: the transition table.

CGS Example



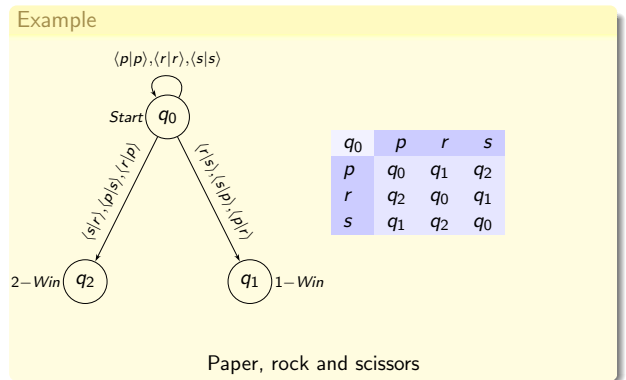
semantics

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

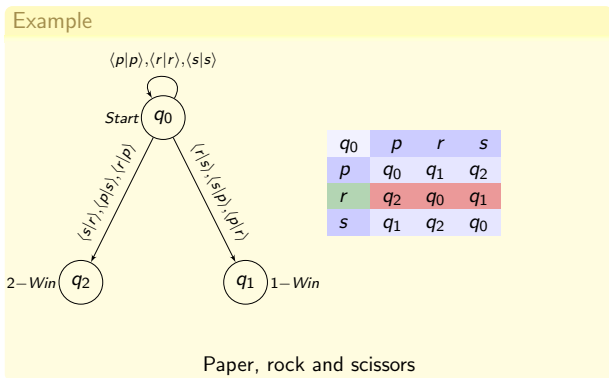
Notations:

- $\text{Next}(\ell) = \{\rightarrow(\ell, \dots, m_{A_i}, \dots) \mid \forall m_{A_i}, 1 \leq i \leq k\}$
- $\text{Next}(\ell, A_j, m) = \{\rightarrow(\ell, \dots, m_{A_{j-1}}, m, m_{A_{j+1}}, \dots)\}$

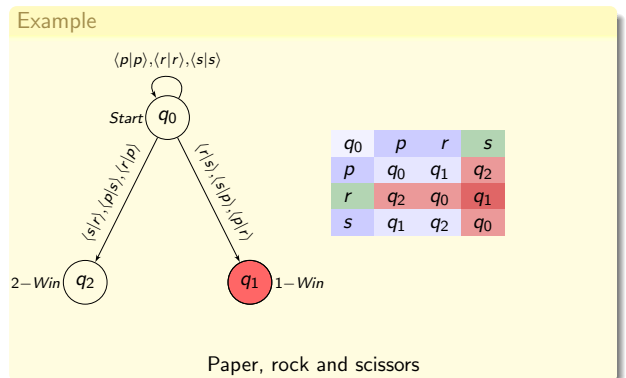
CGS example



CGS example



CGS example



Definition

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

Definition (Alur, Henzinger & Kupferman, 1997)

The syntax of **ATL** is defined by the following grammar:

$$\begin{aligned} \text{ATL} \ni \varphi_s, \psi_s &::= P \mid \neg \varphi_s \mid \varphi_s \vee \psi_s \mid \langle\langle A \rangle\rangle \varphi_p \\ \varphi_p &::= \mathbf{X} \varphi_s \mid \mathbf{G} \varphi_s \mid \varphi_s \mathbf{U} \psi_s. \end{aligned}$$

with $P \in \text{AP}$ and $A \subseteq \text{Agt}$.

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

Semantics

Definition

$$\begin{aligned} \ell \models \langle\langle A \rangle\rangle \varphi_p &\text{ iff } \exists f_A \in \text{Strat}(A). \forall \rho \in \text{Out}(\ell, f_A). \rho \models \varphi_p \\ \rho \models \varphi_s \mathbf{U} \psi_s &\text{ iff } \exists i. \rho[i] \models \psi_s \text{ and } \forall 0 \leq j < i. \rho[j] \models \varphi_s \end{aligned}$$

- Abbreviation: $\Box[A]\varphi$ for $\neg \langle\langle A \rangle\rangle \neg \varphi$
- $\neg \langle\langle A \rangle\rangle \varphi \not\equiv \langle\langle \text{Agt} \setminus A \rangle\rangle \neg \varphi$

Until vs. Weak Until

Definition

- $\varphi \mathbf{W} \psi \equiv \varphi \mathbf{U} \psi \vee \mathbf{G} \varphi$
- $\neg(\varphi \mathbf{U} \psi) \equiv (\neg \psi) \mathbf{W} (\neg \varphi \wedge \neg \psi)$

Theorem

$$\mathbf{E}\varphi \mathbf{W} \psi \equiv \mathbf{E}\mathbf{G} \varphi \vee \mathbf{E}\varphi \mathbf{U} \psi$$

Theorem (Laroussinie, Markey & Oreiby, 2006)

$\langle\langle A \rangle\rangle (a \mathbf{W} b)$ cannot be expressed in ATL.

$$\langle\langle A \rangle\rangle (a \mathbf{W} b) \not\equiv \langle\langle A \rangle\rangle \mathbf{G} a \vee \langle\langle A \rangle\rangle a \mathbf{U} b$$

Until vs. Weak Until

Definition

- $\varphi \mathbf{W} \psi \equiv \varphi \mathbf{U} \psi \vee \mathbf{G} \varphi$
- $\neg(\varphi \mathbf{U} \psi) \equiv (\neg \psi) \mathbf{W} (\neg \varphi \wedge \neg \psi)$

Theorem

$$\mathbf{E}\varphi \mathbf{W} \psi \equiv \mathbf{E}\mathbf{G} \varphi \vee \mathbf{E}\varphi \mathbf{U} \psi$$

Theorem (Laroussinie, Markey & Oreiby, 2006)

$\langle\langle A \rangle\rangle (a \mathbf{W} b)$ cannot be expressed in ATL.

$$\langle\langle A \rangle\rangle (a \mathbf{W} b) \not\equiv \langle\langle A \rangle\rangle \mathbf{G} a \vee \langle\langle A \rangle\rangle a \mathbf{U} b$$

Until vs. Weak Until

Definition

- $\varphi \mathbf{W} \psi \equiv \varphi \mathbf{U} \psi \vee \mathbf{G} \varphi$
- $\neg(\varphi \mathbf{U} \psi) \equiv (\neg \psi) \mathbf{W} (\neg \varphi \wedge \neg \psi)$

Theorem

$$\mathbf{E}\varphi \mathbf{W} \psi \equiv \mathbf{E}\mathbf{G} \varphi \vee \mathbf{E}\varphi \mathbf{U} \psi$$

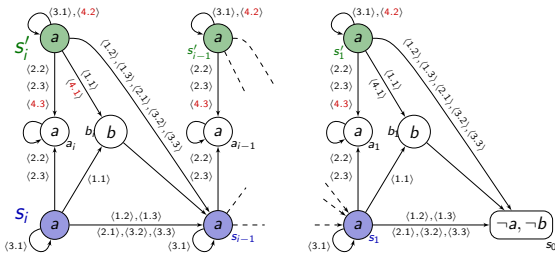
Theorem (Laroussinie, Markey & Oreiby, 2006)

$\langle\langle A \rangle\rangle (a \mathbf{W} b)$ cannot be expressed in ATL.

$$\langle\langle A \rangle\rangle (a \mathbf{W} b) \not\equiv \langle\langle A \rangle\rangle \mathbf{G} a \vee \langle\langle A \rangle\rangle a \mathbf{U} b$$

Proof

$s'_i \models \langle\langle A \rangle\rangle a \mathbf{W} b$ but $s_i \not\models \langle\langle A \rangle\rangle a \mathbf{W} b$



Lemma

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