Temporal Logic with Forgettable Past

F. Laroussinie, N. Markey, Ph. Schnoebelen

{fl,markey,phs}@lsv.ens-cachan.fr

LSV, ENS de Cachan & CNRS UMR8643

Introduction

- Temporal logics are nice specification languages.
  "Any problem is followed eventually by an alarm"

$$\mathsf{G}(\,\mathsf{problem} \;\Rightarrow\; \mathsf{F}\,\mathsf{alarm}\,)$$

Introduction

- Temporal logics are nice specification languages.
  "Any problem is followed eventually by an alarm"

$$\mathsf{G}(\,\mathsf{problem} \;\Rightarrow\; \mathsf{F}\,\mathsf{alarm}\,)$$

- Past operators make specification easier to write.
  "Whenever the alarm rings, then there **has been** some problem in the past"

$$\mathsf{G}(\,\mathsf{alarm} \;\Rightarrow\; \mathsf{F}^{-1}\,\mathsf{problem}\,)$$

Introduction

- Temporal logics are nice specification languages.
  "Any problem is followed eventually by an alarm"

$$\mathsf{G}(\ \text{problem} \ \Rightarrow \ \mathsf{F}\ \text{alarm}\ )$$

- Past operators make specification easier to write.
  "Whenever the alarm rings, then there **has been** some problem in the past"

$$\mathsf{G}(\ \text{alarm} \ \Rightarrow \ \mathsf{F}^{-1}\ \text{problem}\ )$$

We can express this property without F^{-1}:

$$\neg \Big((\neg\ \text{problem})\ \mathsf{U}\ (\text{alarm} \ \wedge \ \neg\ \text{problem}\ ) \Big)$$

Forgettable past

Sometimes it is useful to *forget* the past.
Assume the alarm has a reset button.

Forgettable past

Sometimes it is useful to *forget* the past.
Assume the alarm has a reset button.

"**After a reset**, if the alarm rings, then there has been some problem in the past"

$$\mathsf{G}\left(\,\mathsf{reset} \;\Rightarrow\; \mathsf{G}\,(\,\mathsf{alarm} \;\Rightarrow\; \mathsf{F}^{-1}\,\mathsf{problem}\,)\right)$$

Is it the intended specification ?

Forgettable past

Sometimes it is useful to *forget* the past.
Assume the alarm has a reset button.

"**After a reset**, if the alarm rings, then there has been some problem in the past"

$$\mathsf{G}\left(\mathsf{reset} \Rightarrow \mathsf{G}\,(\mathsf{alarm} \Rightarrow \mathsf{F}^{-1}\,\mathsf{problem}\,)\right)$$

Is it the intended specification ? no !

We do not want to take problems that occured before the reset into account.
We want the reset to also reset the past.

Forgettable past

Sometimes it is useful to *forget* the past.
Assume the alarm has a reset button.

"**After a reset**, if the alarm rings, then there has been some problem in the past"

$$\mathsf{G}\left(\mathsf{reset} \Rightarrow \mathsf{G}\,(\mathsf{alarm} \Rightarrow \mathsf{F}^{-1}\,\mathsf{problem}\,)\right)$$

Is it the intended specification ? no !

We do not want to take problems that occured before the reset into account.
We want the reset to also reset the past.

The N modality ("from Now on") allows us to forget these past states:

$$\mathsf{G}\left(\mathsf{reset} \Rightarrow \mathsf{N}\,\mathsf{G}\,(\mathsf{alarm} \Rightarrow \mathsf{F}^{-1}\,\mathsf{problem}\,)\right)$$

Outline

- Definitions
- Expressive power
  - Comparing expressive power of LTL and LTL+Past
  - of LTL+Past and LTL+Past+Now
- Decision procedures
  - Satisfiability, model checking
  - Complexity
  - Model checking a path
- Conclusion

Definition

$NLTL$ formulae are built from:

- atomic propositions (For ex. problem, alarm),
- boolean combinators ($\wedge$, $\vee$, $\neg$)

Definition

$NLTL$ formulae are built from:

- atomic propositions (For ex. problem, alarm),
- boolean combinators ($\wedge$, $\vee$, $\neg$)
- future modalities: X, U (LTL)

Definition

$NLTL$ formulae are built from:

- atomic propositions (For ex. problem, alarm),
- boolean combinators ($\wedge$, $\vee$, $\neg$)
- future modalities: X, U
- past modalities: X^{-1}, S $(PLTL)$

Definition

$NLTL$ formulae are built from:

- atomic propositions (For ex. problem, alarm),
- boolean combinators ($\wedge$, $\vee$, $\neg$)
- future modalities: X, U
- past modalities: X^{-1}, S
- modality N

Definition

$NLTL$ formulae are built from:

- atomic propositions (For ex. problem, alarm),
- boolean combinators ($\wedge$, $\vee$, $\neg$)
- future modalities: X, U
- past modalities: X^{-1}, S
- modality N

- **plus** all the standard abbreviations:

$$\begin{aligned} \mathsf{F}\,\varphi &\stackrel{\text{def}}{=} \top\,\mathsf{U}\,\varphi & \mathsf{G}\,\varphi &\stackrel{\text{def}}{=} \neg\,\mathsf{F}\,\neg\,\varphi \\ \mathsf{F}^{-1}\,\varphi &\stackrel{\text{def}}{=} \top\,\mathsf{S}\,\varphi & \mathsf{G}^{-1}\,\varphi &\stackrel{\text{def}}{=} \neg\,\mathsf{F}^{-1}\,\neg\,\varphi \end{aligned}$$

The N operator

$NLTL$ formulae are interpreted over pairs π, i: a position along a labeled run (π, ξ).

Semantics of N:

Suffix $\pi(i)\pi(i+1)\ldots$

$$\pi, i \models \mathsf{N}\,\varphi \qquad \text{iff} \qquad \pi^i, 0 \models \varphi$$

The N operator

$NLTL$ formulae are interpreted over pairs π, i: a position along a labeled run (π, ξ).

Semantics of N:

Suffix $\pi(i)\pi(i+1)\ldots$

$$\pi, i \models \mathsf{N}\,\varphi \qquad \text{iff} \qquad \pi^i, 0 \models \varphi$$

Basic properties:

$$\mathsf{N}(\varphi \lor \psi) \equiv \mathsf{N}\varphi \lor \mathsf{N}\psi \qquad\qquad \mathsf{N}\neg\varphi \equiv \neg\mathsf{N}\varphi$$

$$\mathsf{N}\,\mathsf{X}^{-1}\varphi \equiv \bot \qquad\qquad \mathsf{N}(\varphi\,\mathsf{S}\,\psi) \equiv \mathsf{N}\psi$$

$$\mathsf{N}\varphi \equiv \varphi \qquad \text{if } \varphi \text{ is pure-future}$$

Expressive power

We use two notions of equivalence:

- (global) equivalence: $\equiv$

$$\varphi \equiv \psi \quad \stackrel{\text{def}}{\Leftrightarrow} \quad \forall\,\pi,\ \forall\,i,\ \Big(\pi, i \models \varphi \ \Leftrightarrow\ \pi, i \models \psi\Big)$$

Expressive power

We use two notions of equivalence:

- (global) equivalence: $\equiv$

$$\varphi \equiv \psi \quad \overset{\text{def}}{\Leftrightarrow} \quad \forall \pi, \forall i, \quad \Big(\pi, i \models \varphi \Leftrightarrow \pi, i \models \psi\Big)$$

- initial equivalence: $\equiv_i$

$$\varphi \equiv_i \psi \quad \overset{\text{def}}{\Leftrightarrow} \quad \forall \pi, \quad \Big(\pi, 0 \models \varphi \Leftrightarrow \pi, 0 \models \psi\Big)$$

Expressive power

- It is well known that past operators do not add expressive power:

  "Any $PLTL$ formula is *initially equivalent* to an LTL formula."

  where $PLTL$ is $L(\mathsf{U}, \mathsf{X}, \mathsf{S}, \mathsf{X}^{-1})$ and LTL is $L(\mathsf{U}, \mathsf{X})$. [Kam68, Gab89]

Expressive power

- It is well known that past operators do not add expressive power:

  "Any $PLTL$ formula is *initially equivalent* to an LTL formula."

  where $PLTL$ is $L(\mathsf{U}, \mathsf{X}, \mathsf{S}, \mathsf{X}^{-1})$ and LTL is $L(\mathsf{U}, \mathsf{X})$. [Kam68, Gab89]

- This result is based on the separation property:

  "Any $PLTL$ formula is *equivalent* to a boolean combination of pure-future and pure-past formulae." (Example)

Expressive power

- It is well known that past operators do not add expressive power:

  "Any $PLTL$ formula is *initially equivalent* to an LTL formula."

  where $PLTL$ is $L(\mathsf{U}, \mathsf{X}, \mathsf{S}, \mathsf{X}^{-1})$ and LTL is $L(\mathsf{U}, \mathsf{X})$. [Kam68, Gab89]

- This result is based on the separation property:

  "Any $PLTL$ formula is *equivalent* to a boolean combination of pure-future and pure-past formulae." (Example)

- Then LTL, $PLTL$ and $NLTL$ have the same expressive power.
  (any $\mathsf{N}\varphi$ formula is equivalent to a LTL formula)

Expressive power

- It is well known that past operators do not add expressive power:

  "Any $PLTL$ formula is *initially equivalent* to an LTL formula."

  where $PLTL$ is $L(\mathsf{U}, \mathsf{X}, \mathsf{S}, \mathsf{X}^{-1})$ and LTL is $L(\mathsf{U}, \mathsf{X})$. [Kam68, Gab89]

- This result is based on the separation property:

  "Any $PLTL$ formula is *equivalent* to a boolean combination of pure-future and pure-past formulae." (Example)

- Then LTL, $PLTL$ and $NLTL$ have the same expressive power.
  (any $\mathsf{N}\varphi$ formula is equivalent to a LTL formula)

What about succinctness ?

Expressive power - succinctness

Theorem: $PLTL$ can be exponentially more succinct than LTL.

Expressive power - succinctness

Theorem: $PLTL$ can be exponentially more succinct than LTL.

Proof: Let $\{p_0, p_1, \ldots, p_n\}$ be a set of atomic propositions.

Expressive power - succinctness

Theorem: $PLTL$ can be exponentially more succinct than LTL.

Proof: Let $\{p_0, p_1, \ldots, p_n\}$ be a set of atomic propositions.

- The property "any two future states that agree on $p_1, \ldots, p_n$ also agree on p_0" can only be expressed by $PLTL$ (or LTL) formulae of size at least $\Omega(2^n)$. [EVW97].

Expressive power - succinctness

Theorem: $PLTL$ can be exponentially more succinct than LTL.

Proof: Let $\{p_0, p_1, \ldots, p_n\}$ be a set of atomic propositions.

- The property "any two future states that agree on $p_1, \ldots, p_n$ also agree on p_0" can only be expressed by $PLTL$ (or LTL) formulae of size at least $\Omega(2^n)$. [EVW97].

- The $PLTL$ formula

$$\Phi \stackrel{\text{def}}{=} \mathsf{G}\Big[\Big(\bigwedge_{i=1}^{n} \big(p_i \Leftrightarrow \mathsf{F}^{-1}(\neg \mathsf{X}^{-1}\top \wedge p_i)\big)\Big) \Rightarrow \Big(p_0 \Leftrightarrow \mathsf{F}^{-1}(\neg \mathsf{X}^{-1}\top \wedge p_0)\Big)\Big]$$

states that "any future state that agrees with the initial state on $p_1, \ldots, p_n$ also agrees on p_0". Let Ψ be an LTL formula initially equivalent to Φ.

Expressive power - succinctness

Theorem: $PLTL$ can be exponentially more succinct than LTL.

Proof: Let $\{p_0, p_1, \ldots, p_n\}$ be a set of atomic propositions.

- The property "any two future states that agree on $p_1, \ldots, p_n$ also agree on p_0" can only be expressed by $PLTL$ (or LTL) formulae of size at least $\Omega(2^n)$. [EVW97].

- The $PLTL$ formula

$$\Phi \stackrel{\text{def}}{=} \mathsf{G}\Big[\Big(\bigwedge_{i=1}^{n}\big(p_i \Leftrightarrow \mathsf{F}^{-1}(\neg \mathsf{X}^{-1}\top \wedge p_i)\big)\Big) \Rightarrow \Big(p_0 \Leftrightarrow \mathsf{F}^{-1}(\neg \mathsf{X}^{-1}\top \wedge p_0)\Big)\Big]$$

  states that "any future state that agrees with the initial state on $p_1, \ldots, p_n$ also agrees on p_0". Let Ψ be an LTL formula initially equivalent to Φ.

- Therefore $(\mathsf{G}\,\Psi)$ expresses the first property and $|\Psi|$ is in $\Omega(2^n)$!

Expressive power - succinctness

Theorem: $NLTL$ can be exponentially more succinct than $PLTL$.

Expressive power - succinctness

Theorem: $NLTL$ can be exponentially more succinct than $PLTL$.

Proof: Let $\{p_0, p_1, \ldots, p_n\}$ be a set of atomic propositions.
We still write

$$\Phi = \mathsf{G}\Big[\Big(\bigwedge_{i=1}^{n}\big(p_i \Leftrightarrow \mathsf{F}^{-1}(\neg\mathsf{X}^{-1}\top \wedge p_i)\big)\Big) \Rightarrow \Big(p_0 \Leftrightarrow \mathsf{F}^{-1}(\neg\mathsf{X}^{-1}\top \wedge p_0)\Big)\Big]$$

The $NLTL$ formula $\mathsf{G}\,\mathsf{N}\,\Phi$ clearly states that "any two future states that agree on $p_1, \ldots, p_n$ also agree on p_0".
Then any equivalent $PLTL$ formula has a size $\Omega(2^n)$.

Verification problems

We are interested in:

- Satisfiability: Given ϕ, is there some π, i such that: $\pi, i \models \phi$?
- Initial satisfiability: Given ϕ, is there some π such that: $\pi, 0 \models \phi$?
- Model checking: Given ϕ and a Kripke structure K, do we have, for any run π of K: $\pi, 0 \models \phi$?
- ...

Deciding satisfiability for linear time TL

- Let Φ be an *LTL* or *PLTL* formula. [VW94]
  Build a Büchi automaton $\mathcal{A}_\Phi = \langle Q, \rightarrow, F \rangle$ with $Q = 2^{\mathsf{SubF}(\Phi)}$.
  Φ is satisfiable $\Leftrightarrow$ there exists an accepting run in $\mathcal{A}_\Phi$.

$$\left.\begin{array}{l} |\mathcal{A}_\Phi| \text{ is in } O(2^{|\Phi|}) \\ \text{Emptiness in BA is NLOGSPACE-complete} \end{array}\right\} \Rightarrow \text{Algorithm in PSPACE.}$$

Deciding satisfiability for linear time TL

- Let Φ be an *LTL* or *PLTL* formula. [VW94]
  Build a Büchi automaton $\mathcal{A}_\Phi = \langle Q, \rightarrow, F \rangle$ with $Q = 2^{\mathsf{SubF}(\Phi)}$.
  Φ is satisfiable $\Leftrightarrow$ there exists an accepting run in $\mathcal{A}_\Phi$.

$$\left.\begin{array}{l} |\mathcal{A}_\Phi| \text{ is in } O(2^{|\Phi|}) \\ \text{Emptiness in BA is NLOGSPACE-complete} \end{array}\right\} \Rightarrow \text{Algorithm in PSPACE.}$$

- Let Φ be an *LTL* formula. [Var94]
  Build an Alternating Büchi automaton $\mathcal{A}_\Phi = \langle Q, \delta, F \rangle$ with $Q = \mathsf{SubF}(\Phi)$.
  Φ is satisfiable $\Leftrightarrow$ there exists an accepting run in $\mathcal{A}_\Phi$.

$$\left.\begin{array}{l} |\mathcal{A}_\Phi| \text{ is in } O(|\Phi|) \\ \text{Emptiness for ABA is PSPACE-complete} \end{array}\right\} \Rightarrow \text{Algorithm in PSPACE.}$$

Deciding satisfiability for linear time TL

- Let Φ be an LTL or $PLTL$ formula. [VW94]
  Build a Büchi automaton $\mathcal{A}_\Phi = \langle Q, \rightarrow, F\rangle$ with $Q = 2^{\mathsf{SubF}(\Phi)}$.
  Φ is satisfiable $\Leftrightarrow$ there exists an accepting run in $\mathcal{A}_\Phi$.

$$\left.\begin{array}{l} |\mathcal{A}_\Phi| \text{ is in } O(2^{|\Phi|}) \\ \text{Emptiness in BA is NLOGSPACE-complete} \end{array}\right\} \Rightarrow \text{Algorithm in PSPACE.}$$

- Let Φ be an LTL formula. [Var94]
  Build an Alternating Büchi automaton $\mathcal{A}_\Phi = \langle Q, \delta, F\rangle$ with $Q = \mathsf{SubF}(\Phi)$.
  Φ is satisfiable $\Leftrightarrow$ there exists an accepting run in $\mathcal{A}_\Phi$.

$$\left.\begin{array}{l} |\mathcal{A}_\Phi| \text{ is in } O(|\Phi|) \\ \text{Emptiness for ABA is PSPACE-complete} \end{array}\right\} \Rightarrow \text{Algorithm in PSPACE.}$$

These algorithms are optimal: LTL satisfiability is PSPACE-hard.

Deciding satisfiability for LTL

$\mathsf{SubF}(\Phi)$ = set of Φ subformulae + negations + $\mathsf{X}(\_\,\mathsf{U}\,\_)$.
States are *Atoms*: coherent subsets of $\mathsf{SubF}(\Phi)$

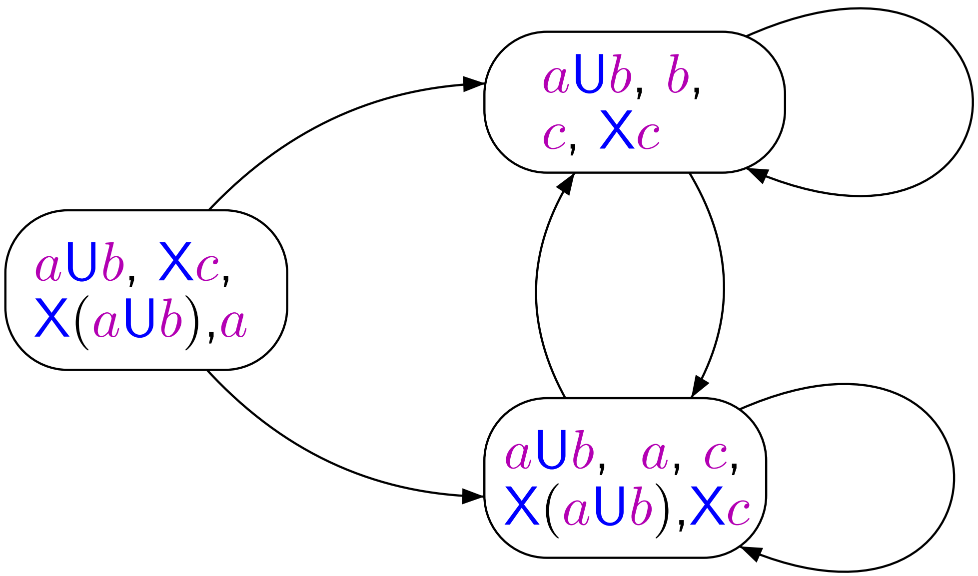
$a\mathsf{U}b$, b,
c, $\mathsf{X}c$

$a\mathsf{U}b$, $\mathsf{X}c$,
$\mathsf{X}(a\mathsf{U}b)$,a

$a\mathsf{U}b$, a, c,
$\mathsf{X}(a\mathsf{U}b)$,$\mathsf{X}c$

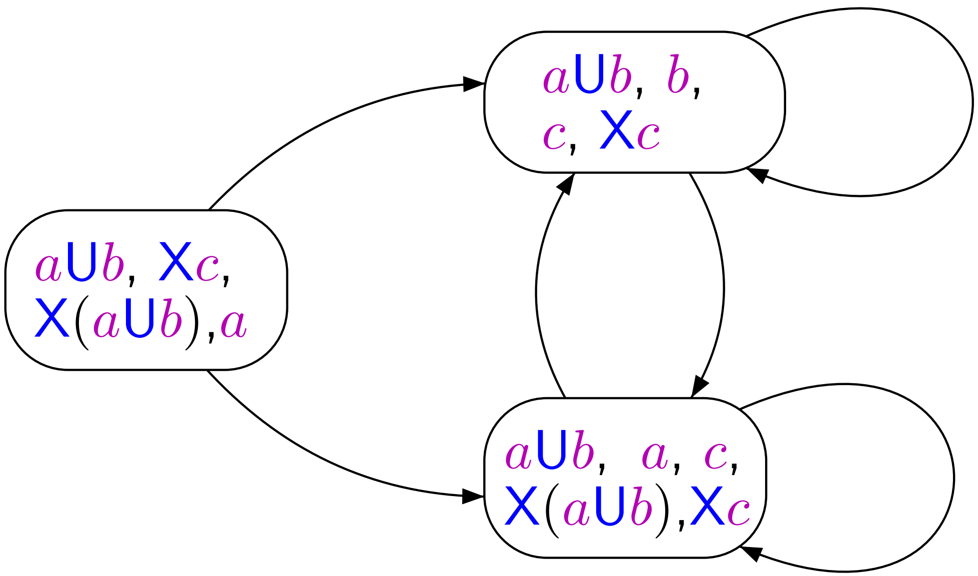
Deciding satisfiability for LTL

SubF(Φ) = set of Φ subformulae + negations + X(_ U _).
States are *Atoms*: coherent subsets of SubF(Φ)



Deciding satisfiability for LTL

SubF(Φ) $=$ set of Φ subformulae $+$ negations $+$ X(_ U _).
States are *Atoms*: coherent subsets of SubF(Φ)

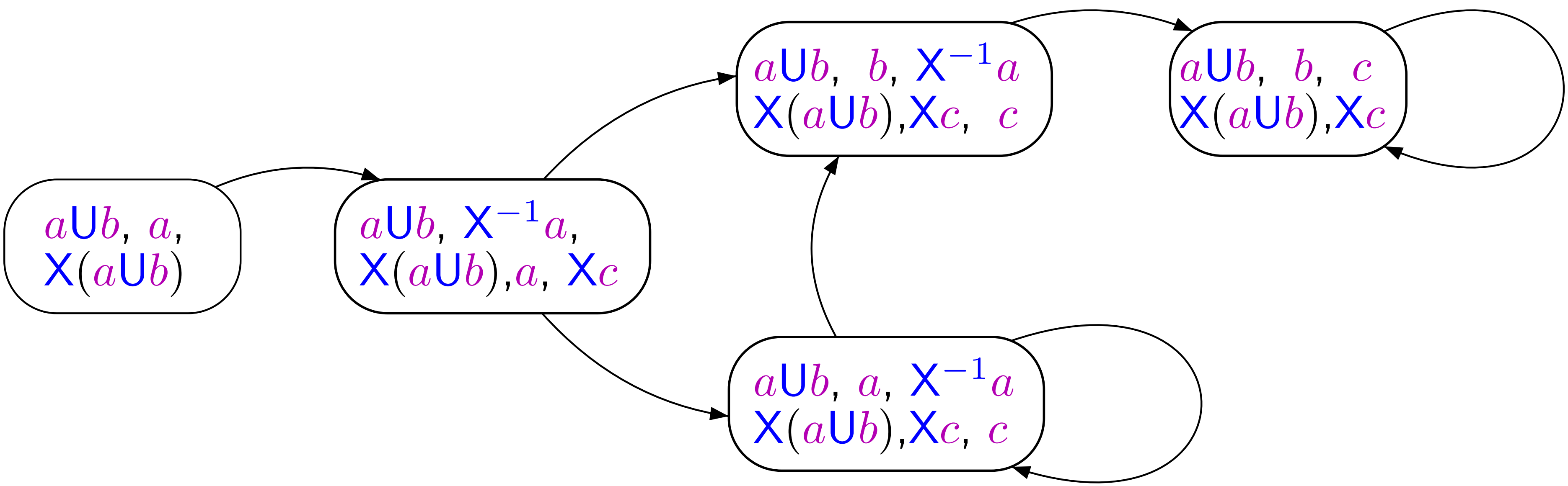


$+$ fairness conditions for Until formulae.

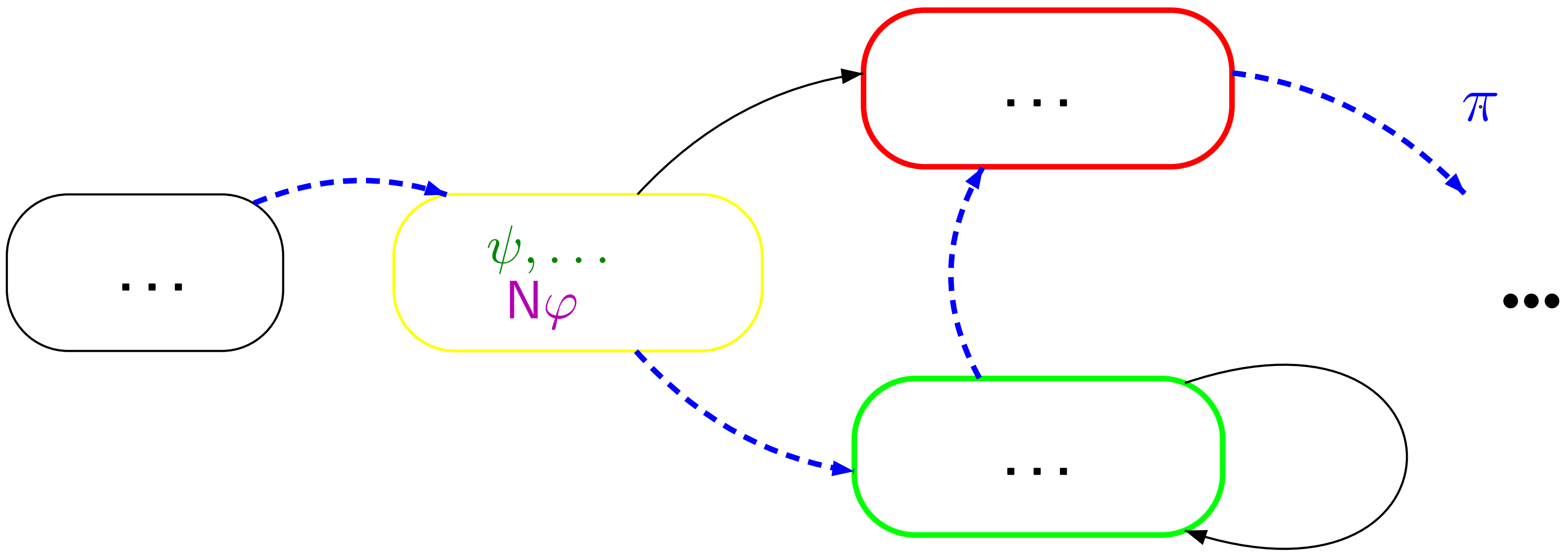
$\varphi \in$ SubF(Φ) is satisfiable **iff** there is an accepting run going through $A \ni \varphi$.

Deciding satisfiability for $PLTL$

$\mathsf{SubF}(\Phi) =$ set of Φ subformulae $+ \ldots +$ $\mathsf{X}^{-1}(\_\,\mathsf{S}\,\_)$ $+$ $\mathsf{X}^{-1}\top$.

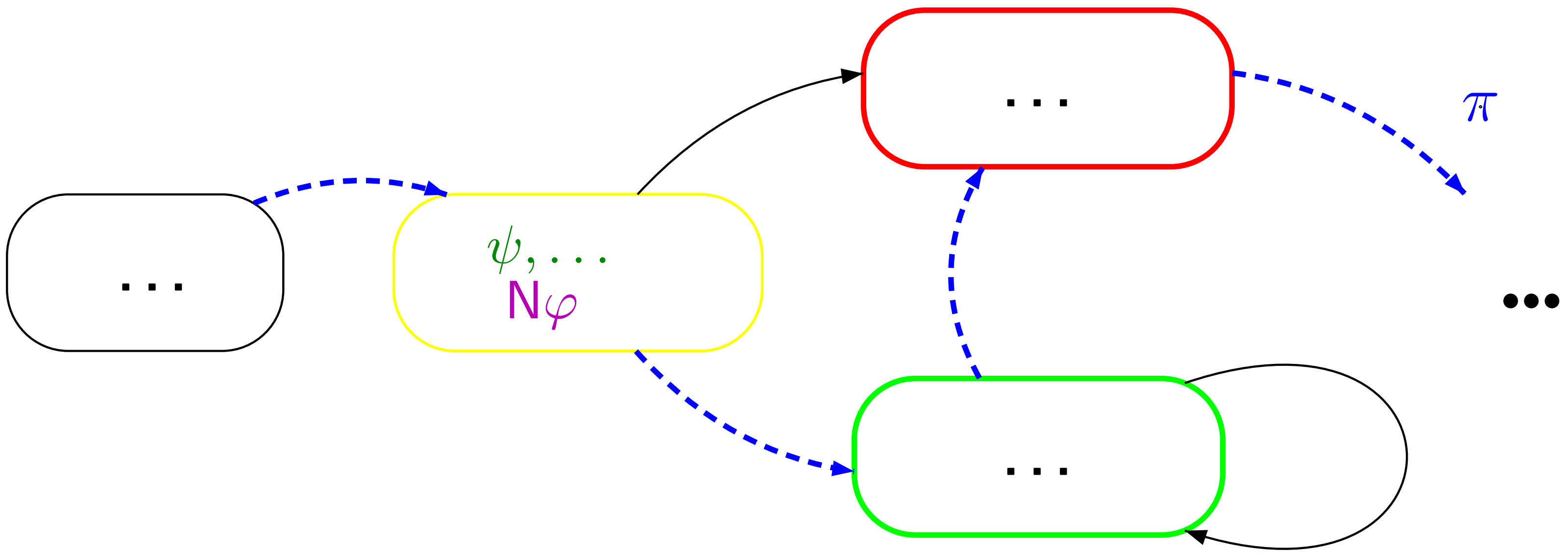


And $NLTL$?



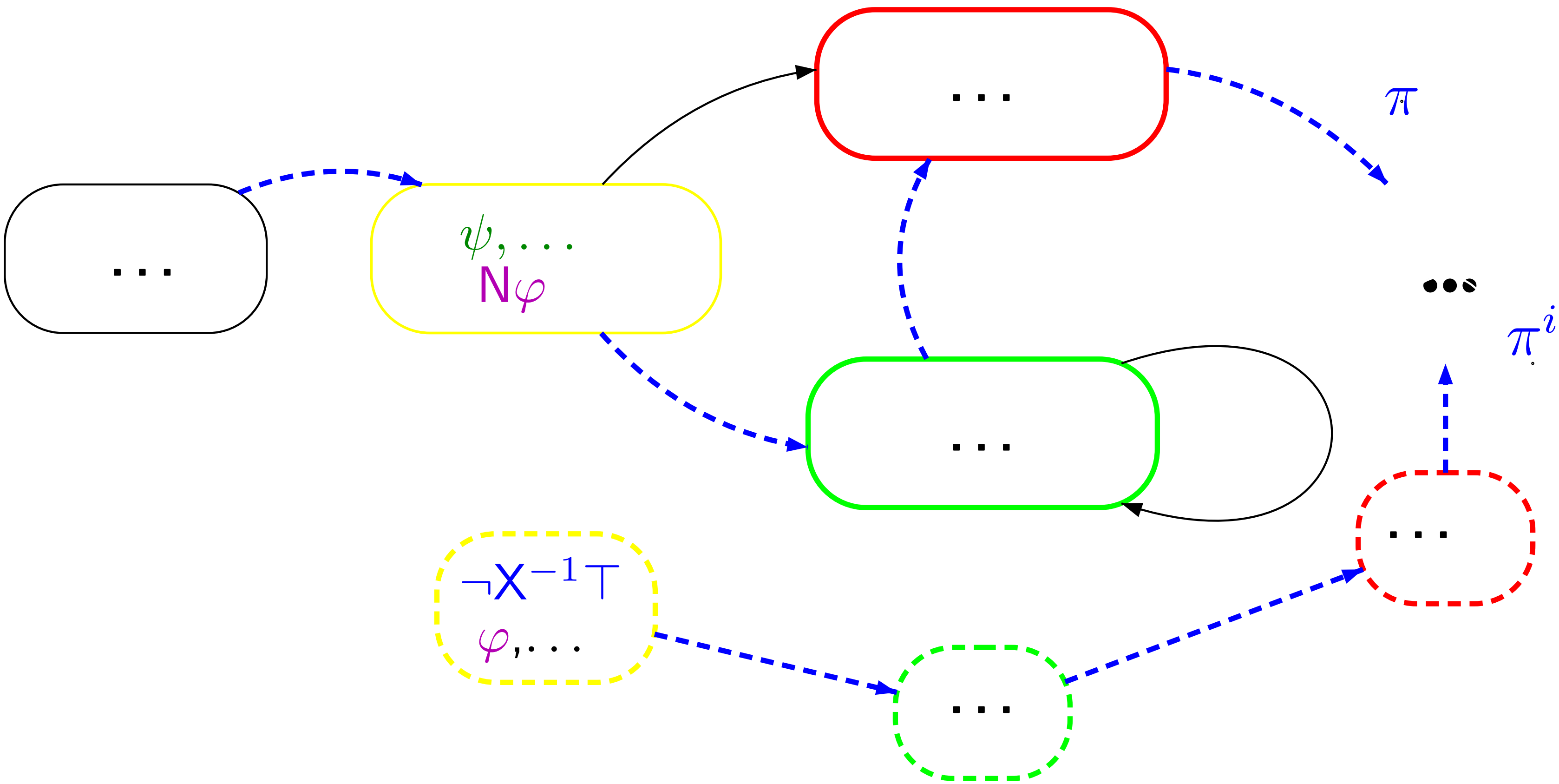
And $NLTL$?

$\pi, i \models \psi$
$\pi^i, 0 \models \varphi$



And *NLTL* **?**

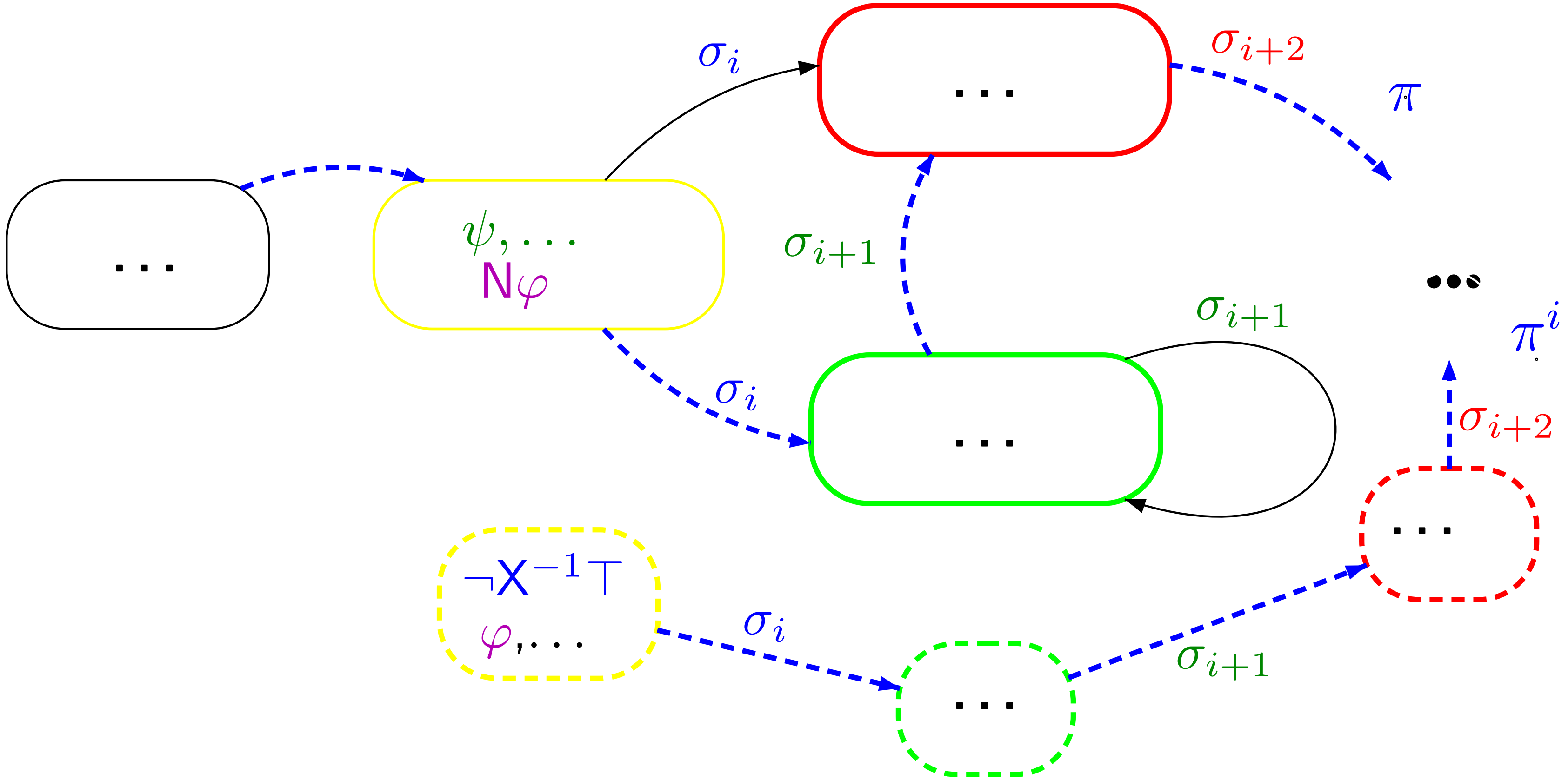
$$\pi, i \models \psi$$
$$\pi^i, 0 \models \varphi$$



And *NLTL* ?

$\pi, i \models \psi$
$\pi^i, 0 \models \varphi$

$\sigma_i \in 2^{AP}$: atomic propositions of $\pi(i)$.



Alternating Büchi automata for $NLTL$ formulae

$\mathcal{A}_\Phi = \langle \Sigma, S, \rho, S_0, \mathcal{F} \rangle$ is defined as follows:

- $\Sigma = 2^{AP}$, $\quad S = \mathsf{Atom}(\Phi)$, $\quad S_0 = \{A \in \mathsf{Atom}(\Phi) \mid \neg \mathsf{X}^{-1}\top \in A\}$,

Alternating Büchi automata for $NLTL$ formulae

$\mathcal{A}_\Phi = \langle \Sigma, S, \rho, S_0, \mathcal{F} \rangle$ is defined as follows:

- $\Sigma = 2^{AP}$, $\quad S = \mathsf{Atom}(\Phi)$, $\quad S_0 = \{A \in \mathsf{Atom}(\Phi) \mid \neg \mathsf{X}^{-1}\top \in A\}$,
- $\rho(A, \sigma) = \bigvee_{A' \in \mathsf{Succ}(A,\sigma)} \left(A' \wedge \bigvee_{A'' \in \mathsf{Now}(A')} A'' \right)$ with:

$$\begin{aligned}\mathsf{Succ}(A, \sigma_A) \stackrel{\text{def}}{=} \{ & A' \in \mathsf{Atom}(\Phi) \mid \mathsf{X}^{-1}\top \in A' \text{ and} \\ & \forall \mathsf{X}\alpha \in \mathsf{SubF}(\Phi),\ \mathsf{X}\alpha \in A \Leftrightarrow \alpha \in A' \text{ and} \\ & \forall \mathsf{X}^{-1}\alpha \in \mathsf{SubF}(\Phi),\ \alpha \in A \Leftrightarrow \mathsf{X}^{-1}\alpha \in A' \}\end{aligned}$$

$$\mathsf{Now}(A') \stackrel{\text{def}}{=} \{A'' \in S_0 \mid \forall \mathsf{N}\alpha \in \mathsf{SubF}(\Phi), \mathsf{N}\alpha \in A' \Leftrightarrow \mathsf{N}\alpha, \alpha \in A''\}$$

Alternating Büchi automata for $NLTL$ formulae

$\mathcal{A}_\Phi = \langle \Sigma, S, \rho, S_0, \mathcal{F} \rangle$ is defined as follows:

- $\Sigma = 2^{AP}$, $\quad S = \mathsf{Atom}(\Phi)$, $\quad S_0 = \{A \in \mathsf{Atom}(\Phi) \mid \neg \mathsf{X}^{-1}\top \in A\}$,
- $\rho(A, \sigma) = \bigvee_{A' \in \mathsf{Succ}(A,\sigma)} \left(A' \wedge \bigvee_{A'' \in \mathsf{Now}(A')} A'' \right)$ with:

$$\begin{aligned}\mathsf{Succ}(A, \sigma_A) \stackrel{\text{def}}{=} \{A' \in \mathsf{Atom}(\Phi) \mid\, & \mathsf{X}^{-1}\top \in A' \text{ and} \\ & \forall \mathsf{X}\alpha \in \mathsf{SubF}(\Phi),\ \mathsf{X}\alpha \in A \Leftrightarrow \alpha \in A' \text{ and} \\ & \forall \mathsf{X}^{-1}\alpha \in \mathsf{SubF}(\Phi),\ \alpha \in A \Leftrightarrow \mathsf{X}^{-1}\alpha \in A'\}\end{aligned}$$

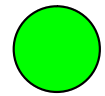
$$\mathsf{Now}(A') \stackrel{\text{def}}{=} \{A'' \in S_0 \mid \forall \mathsf{N}\alpha \in \mathsf{SubF}(\Phi), \mathsf{N}\alpha \in A' \Leftrightarrow \mathsf{N}\alpha, \alpha \in A''\}$$

- $\mathcal{F} = \{F_1, \ldots, F_k\}$ with:

$$F_i \stackrel{\text{def}}{=} \{A \in \mathsf{Atom}(\Phi) \mid \neg \mathsf{X}^{-1}\top \in A \text{ or } \psi_i \in A \text{ or } \neg(\varphi_i \mathsf{U} \psi_i) \in A\}$$

An accepting run π

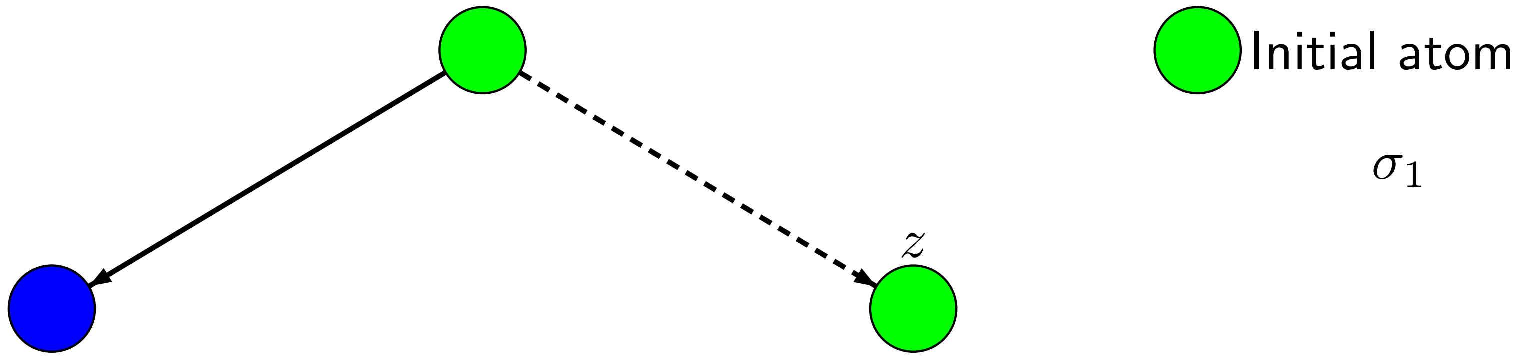
$\xi(\pi(i)) = \sigma_i$



Initial atom

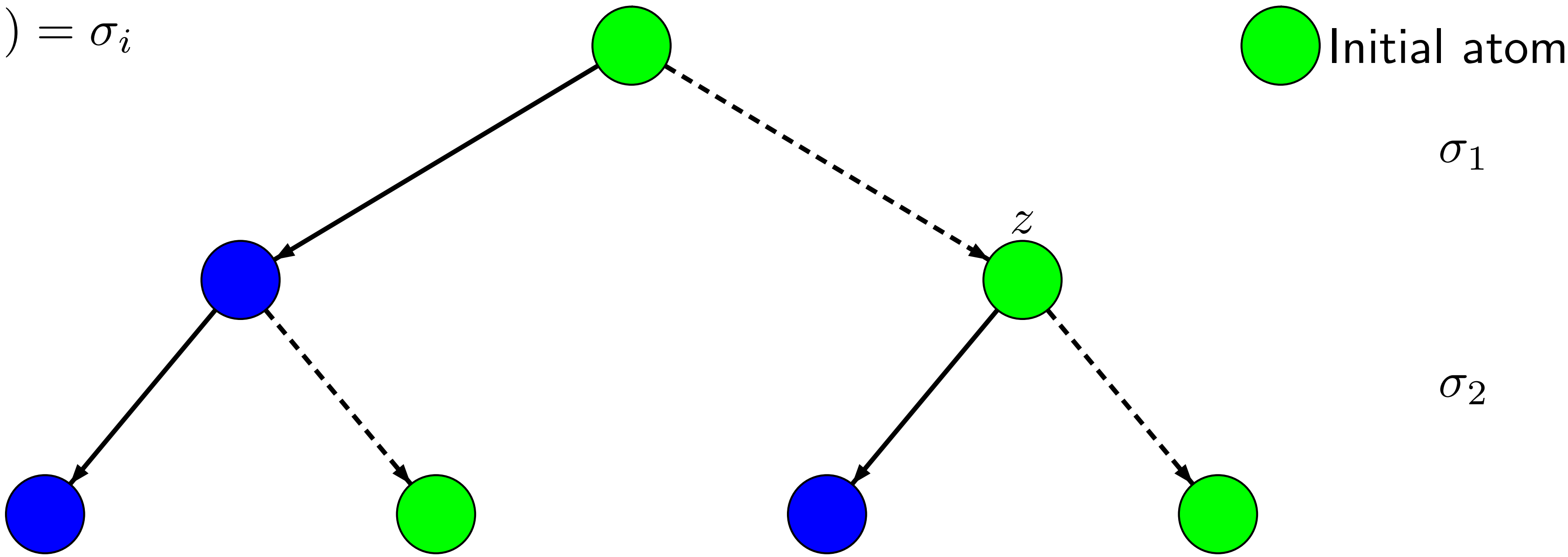
An accepting run π

$\xi(\pi(i)) = \sigma_i$



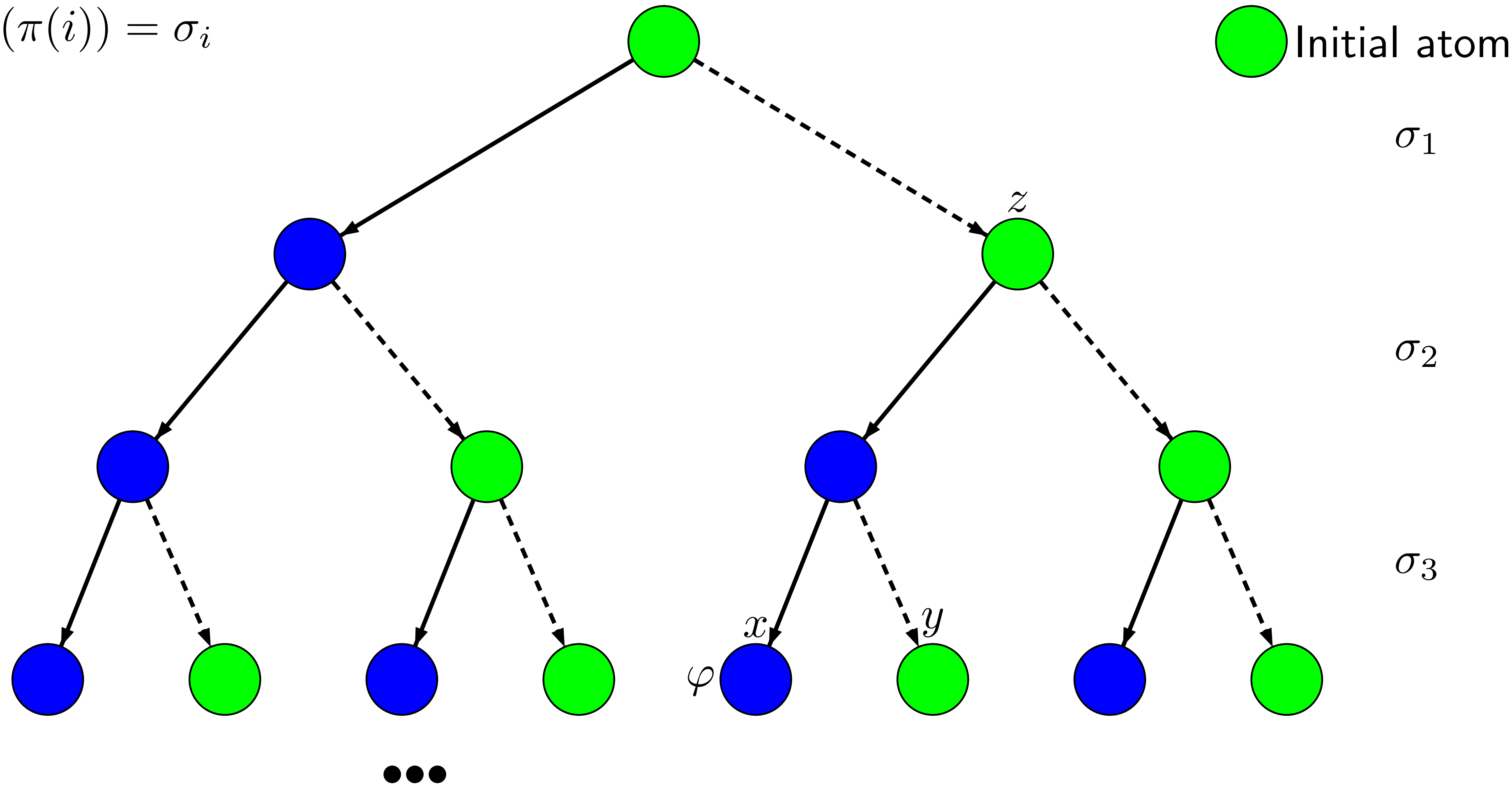
An accepting run π

$\xi(\pi(i)) = \sigma_i$

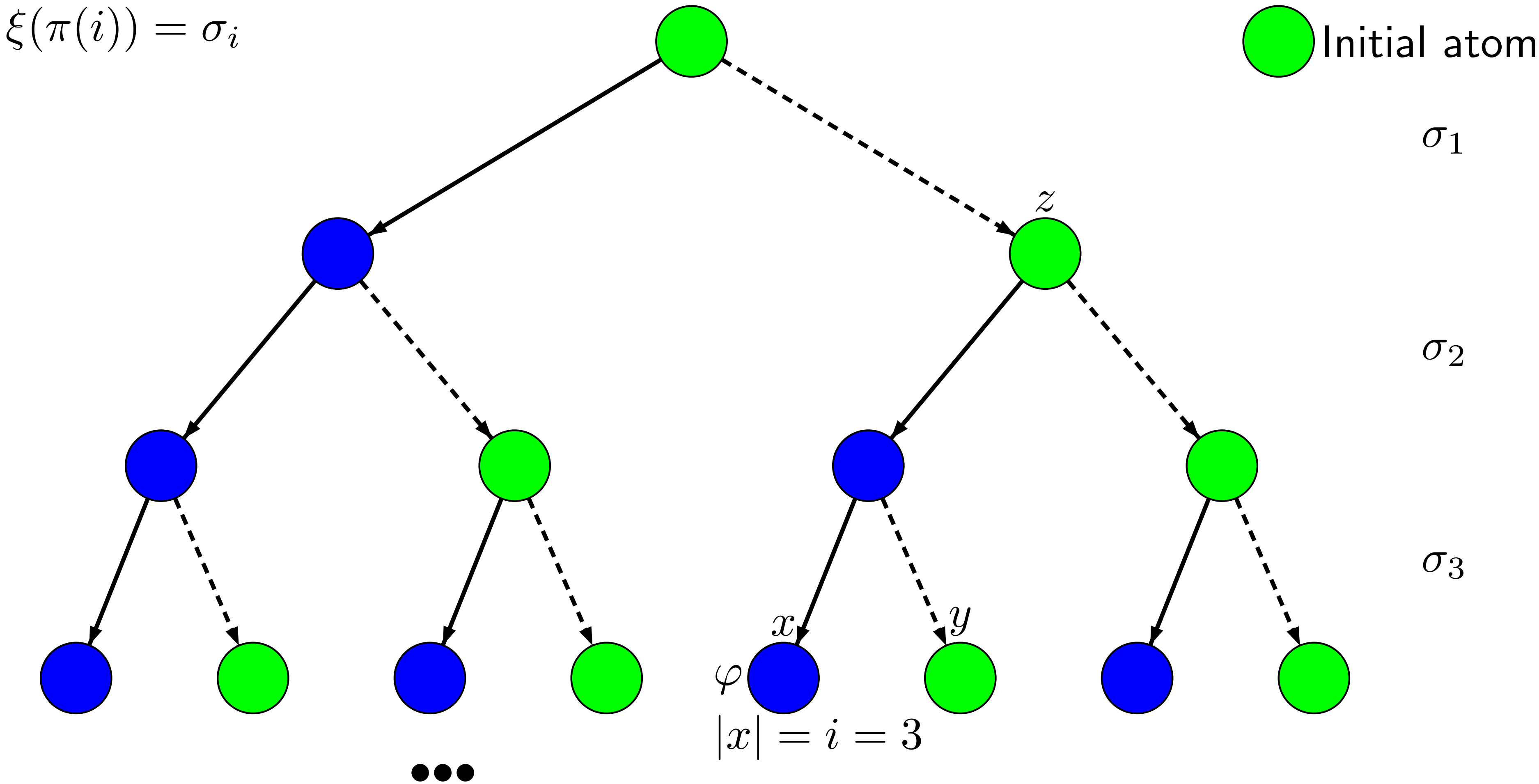


An accepting run π

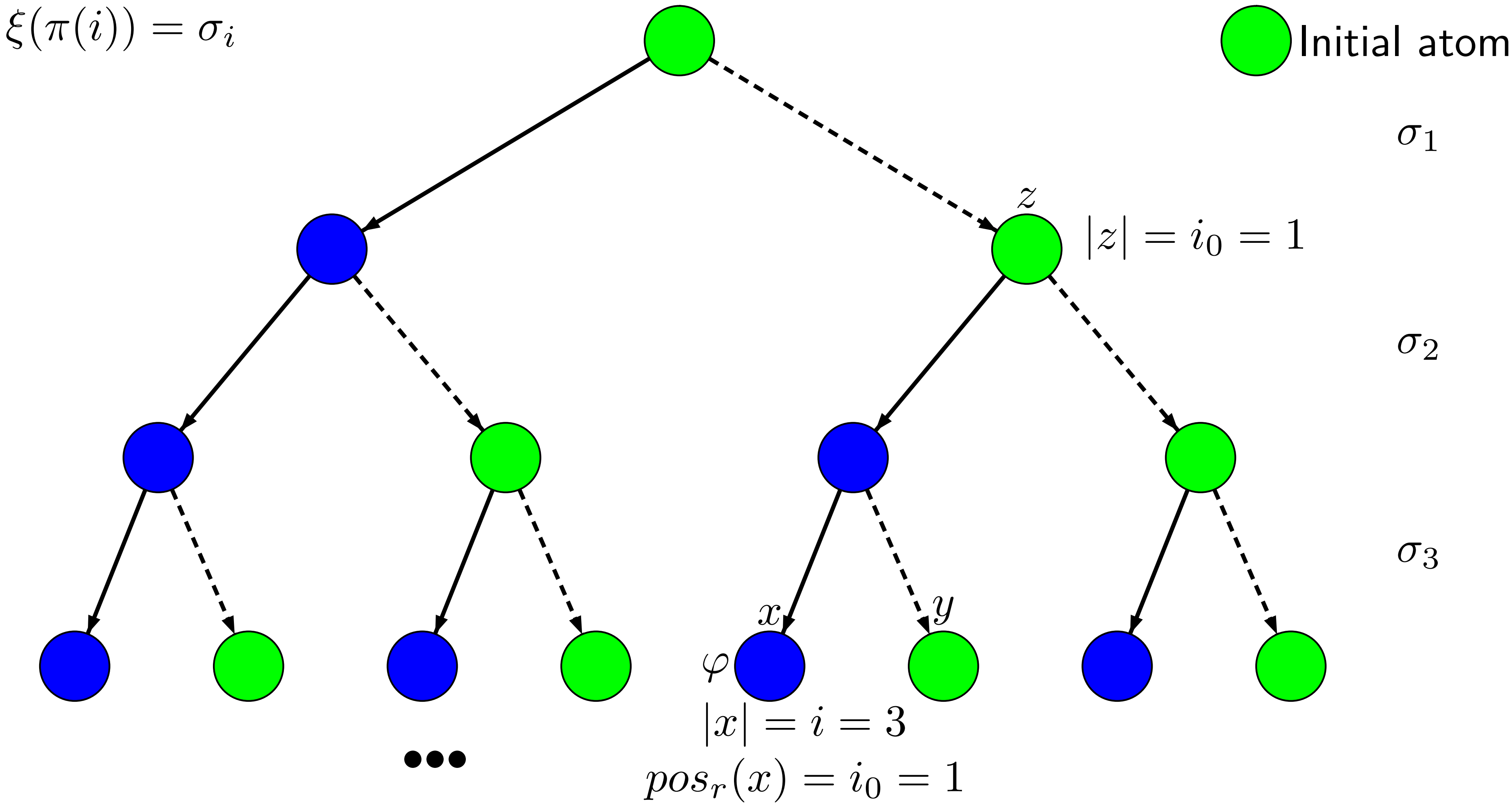
$\xi(\pi(i)) = \sigma_i$



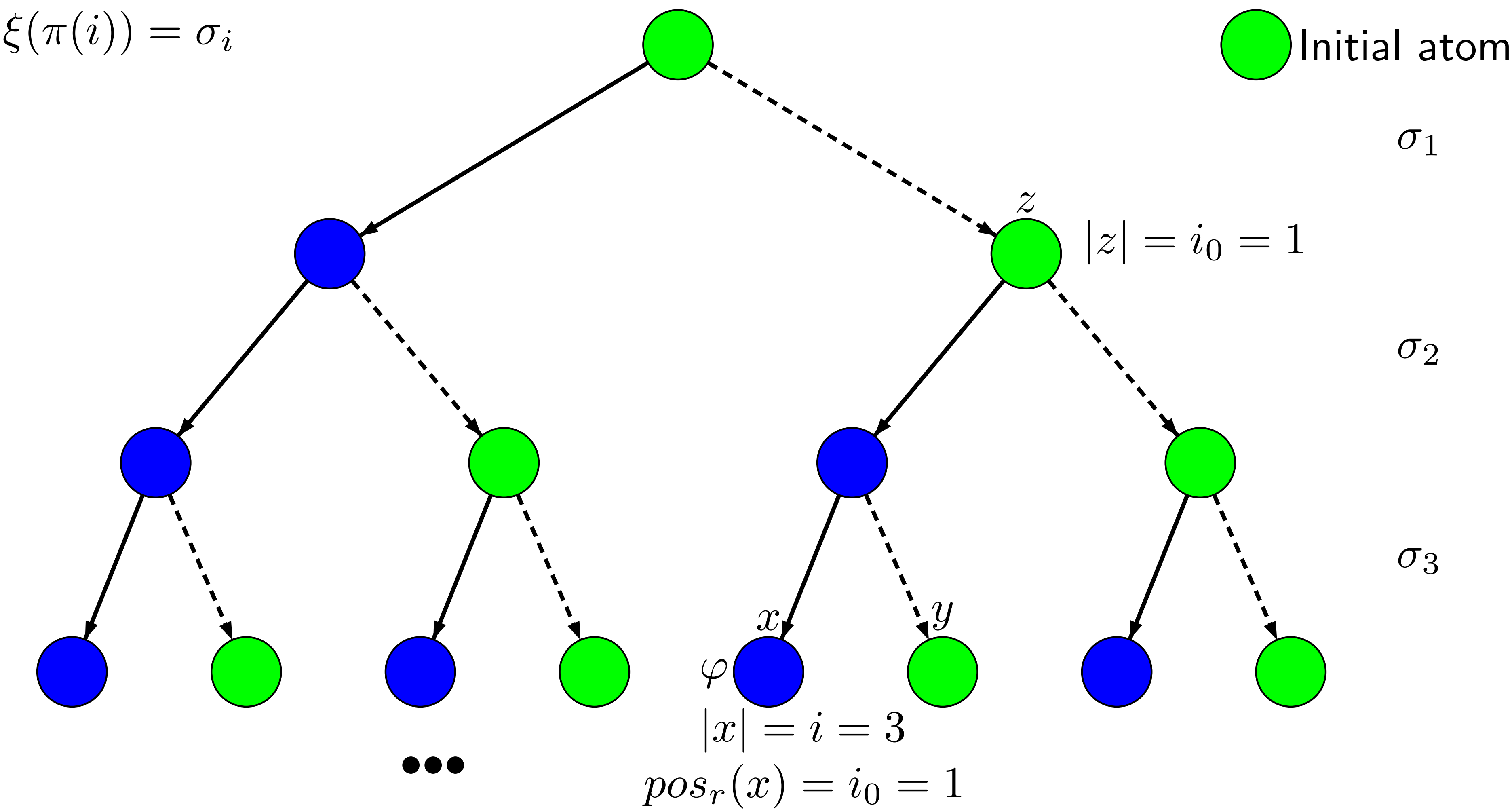
An accepting run π



An accepting run π



An accepting run π



Lemma: $\pi^{i_0}, i - i_0 \models \varphi \quad \Leftrightarrow \quad \exists\, x \; s.t. \left(|x| = i, \quad pos_r(x) = i_0, \quad \varphi \in x \right)$

Alternating Büchi automata for $NLTL$ formulae

Proposition: Let Φ be an $NLTL$ formula, then Φ is satisfiable **iff** there exists an accepting run in $\mathcal{A}_{\mathsf{F}\Phi}$ starting from a node containing $\mathsf{F}\Phi$.

Proposition: Non-emptiness problem of alternating Büchi automaton can be solved in space polynomial in the size of the automaton. [VW94]

Theorem: Satisfiability for $NLTL$ formulae can be decided in EXPSPACE.

Theorem: Model checking Kripke structures for $NLTL$ formulae can be decided in EXPSPACE.

EXPSPACE-hardness of $NLTL$ model checking

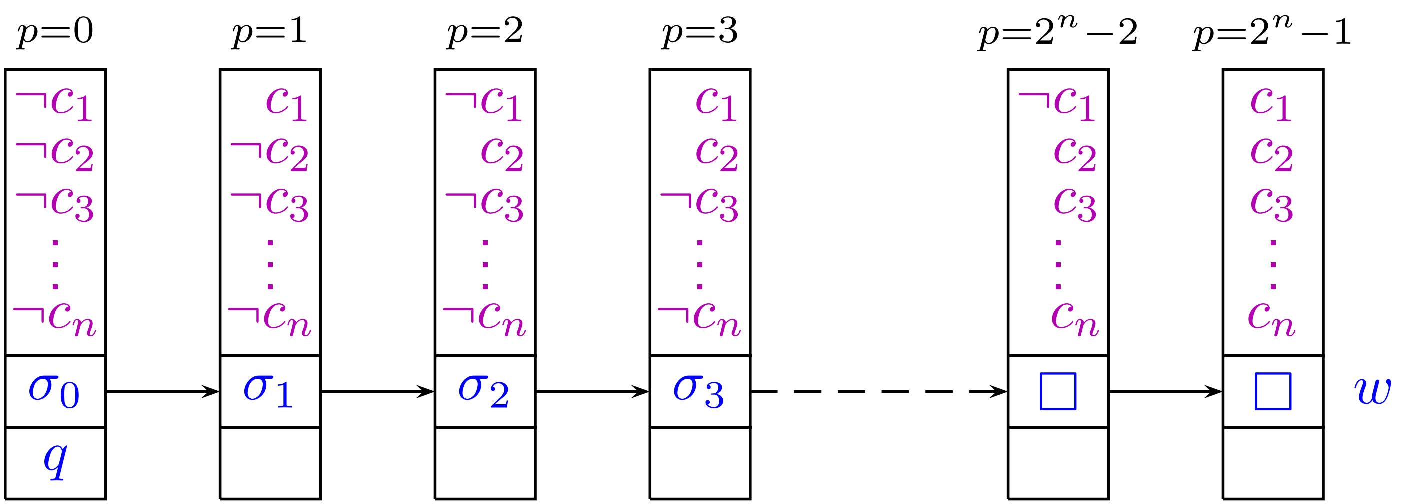
Proposition: Satisfiability and model checking for $NLTL$ are $\mathrm{EXPSPACE}$-hard.

EXPSPACE-hardness of $NLTL$ model checking

Proposition: Satisfiability and model checking for $NLTL$ are EXPSPACE-hard.

Let $\mathcal{M} = \langle \Sigma, Q_{\mathcal{M}}, q_0, q_F, \mathcal{T} \rangle$ be a Turing machine that operates in exponential space. The run of $\mathcal{M}$ on some input word w of length n can be described by an $NLTL$-formula Φ:

- The set AP contains Σ, $Q_{\mathcal{M}}$, and $\{c_1, \ldots, c_n\}$.
- A configuration of $\mathcal{M}$ is a sequence of 2^n states:



EXPSPACE-hardness of $NLTL$ model checking

The formula Φ has to state that:

- each state contains exactly one proposition from Σ;

EXPSPACE-hardness of $NLTL$ model checking

The formula Φ has to state that:

- each state contains exactly one proposition from Σ;
- exactly one control state occurs in each configuration of $\mathcal{M}$;

EXPSPACE-hardness of $NLTL$ model checking

The formula Φ has to state that:

- each state contains exactly one proposition from Σ;
- exactly one control state occurs in each configuration of $\mathcal{M}$;
- the 2^n cells of the tape are all present, in increasing order:

$$\mathsf{G}\Big((c_1 \Leftrightarrow \mathsf{X}\neg c_1) \wedge \bigwedge_{i=2}^{n} \Big((\neg(c_i \Leftrightarrow \mathsf{X}c_i)) \Leftrightarrow (c_{i-1} \wedge \mathsf{X}\neg c_{i-1})\Big)\Big)$$

EXPSPACE-hardness of $NLTL$ model checking

- The transitions $(a, b, c) \to b'$ of $\mathcal{M}$ are respected.

EXPSPACE-hardness of $NLTL$ model checking

- The transitions $(a, b, c) \to b'$ of $\mathcal{M}$ are respected.

  We define

$$\phi_{p=0} \stackrel{\mathsf{def}}{=} \bigwedge_{i=1}^{n} \neg c_i \qquad\qquad \phi_{sv} \stackrel{\mathsf{def}}{=} \bigwedge_{i=1}^{n} (c_i \Leftrightarrow \mathsf{F}^{-1}(\neg \mathsf{X}^{-1}\top \wedge c_i))$$

  and write

$$\mathsf{GN}\Big(\bigwedge_{(a,b,c)\to b'} (a \wedge \mathsf{X}b \wedge \mathsf{X}^2 c) \Rightarrow (\neg\phi_{p=0}\mathsf{U}(\phi_{p=0} \wedge \mathsf{X}(\neg\phi_{p=0}\mathsf{U}(\phi_{sv} \wedge \mathsf{X}b')))\Big)$$

EXPSPACE-hardness of $NLTL$ model checking

- The transitions $(a, b, c) \rightarrow b'$ of $\mathcal{M}$ are respected.

  We define

$$\phi_{p=0} \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} \neg c_i \qquad\qquad \phi_{sv} \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} (c_i \Leftrightarrow \mathsf{F}^{-1}(\neg \mathsf{X}^{-1}\top \wedge c_i))$$

  and write

$$\mathsf{GN}\Big(\bigwedge_{(a,b,c)\rightarrow b'} (a \wedge \mathsf{X}b \wedge \mathsf{X}^2 c) \Rightarrow (\neg\phi_{p=0}\mathsf{U}(\phi_{p=0} \wedge \mathsf{X}(\neg\phi_{p=0}\mathsf{U}(\phi_{sv} \wedge \mathsf{X}b'))\Big)$$

This is to check whether we can apply the transition.

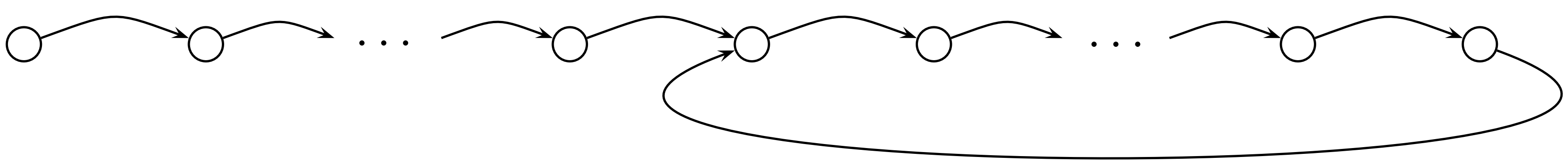
This ensures that we go to the very next configuration.

Here we control that the transition has been correctly applied.

Model checking a path

Theorem: Model checking an $NLTL$-formula ϕ along an ultimately-periodic path can be done in polynomial-time.

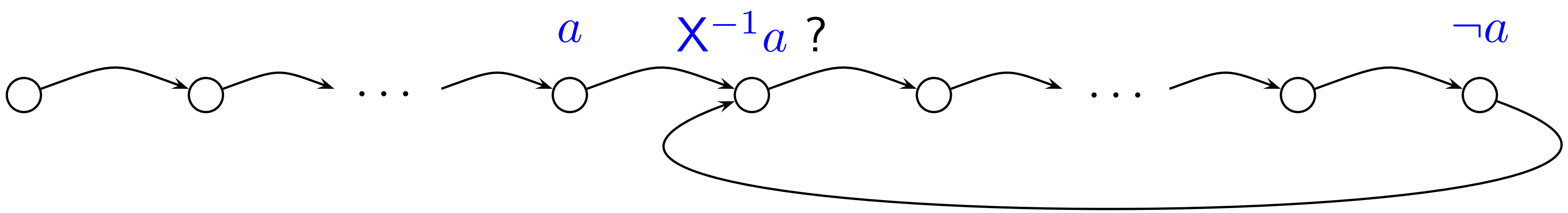
For LTL, we can directly apply CTL algorithm since each state has exactly one successor.



Model checking a path

Theorem: Model checking an $NLTL$-formula ϕ along an ultimately-periodic path can be done in polynomial-time.

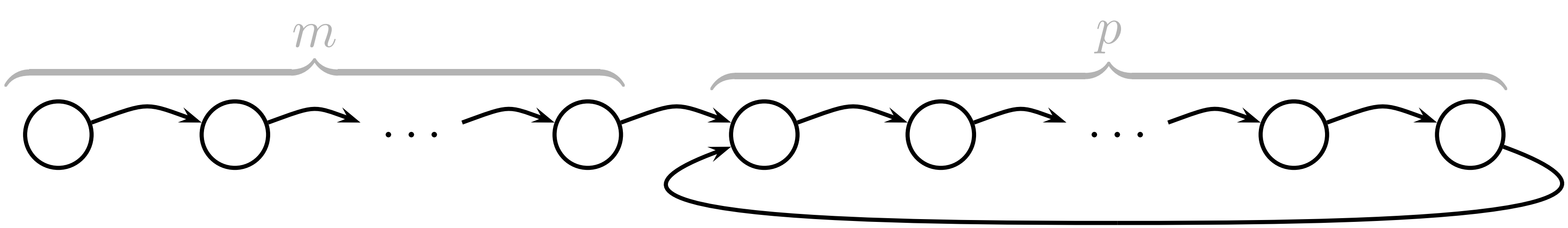
For LTL, we can directly apply CTL algorithm since each state has exactly one successor.



But this does not apply for $PLTL$ because some states don't have exactly one predecessor.

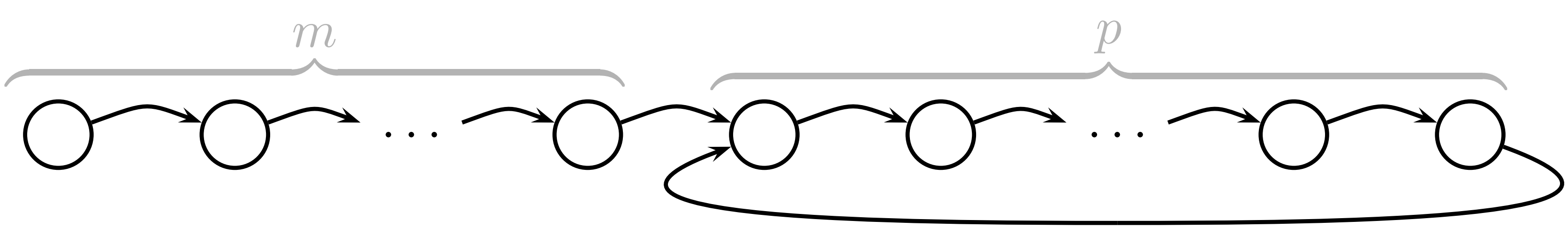
Model checking a path

A *loop of type* (m, p) is an ultimately periodic KS where the initial part has length m and the periodic part has length p.



Model checking a path

A *loop of type* (m, p) is an ultimately periodic KS where the initial part has length m and the periodic part has length p.

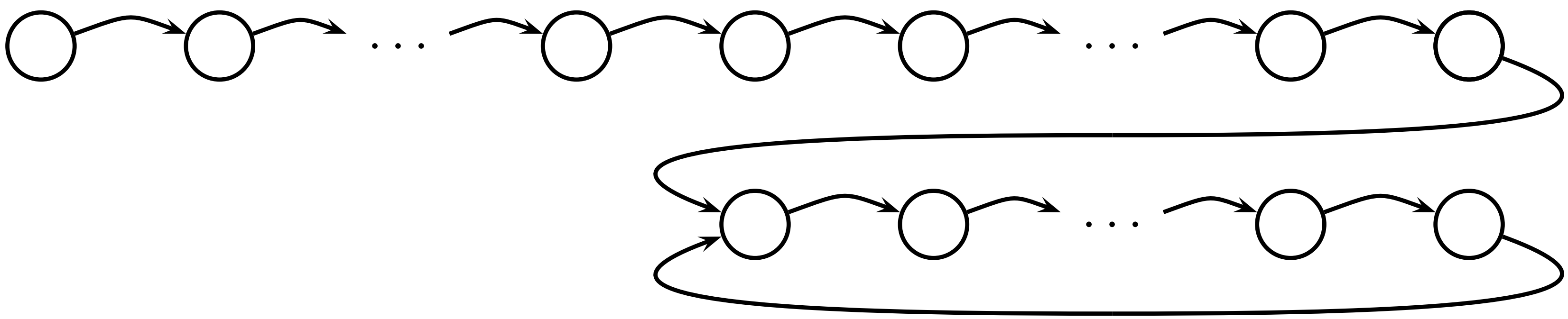


Lemma: For any loop L of type (m, p), for any $NLTL$ formula ϕ with at most $h(\phi)$ nested past-time modalities, and for any $k \geq m + p \cdot h(\phi)$, we have

$$\pi, k \models \phi \qquad \Leftrightarrow \qquad \pi, k + p \models \phi.$$

Model checking a path

A *loop of type* (m, p) is an ultimately periodic KS where the initial part has length m and the periodic part has length p.

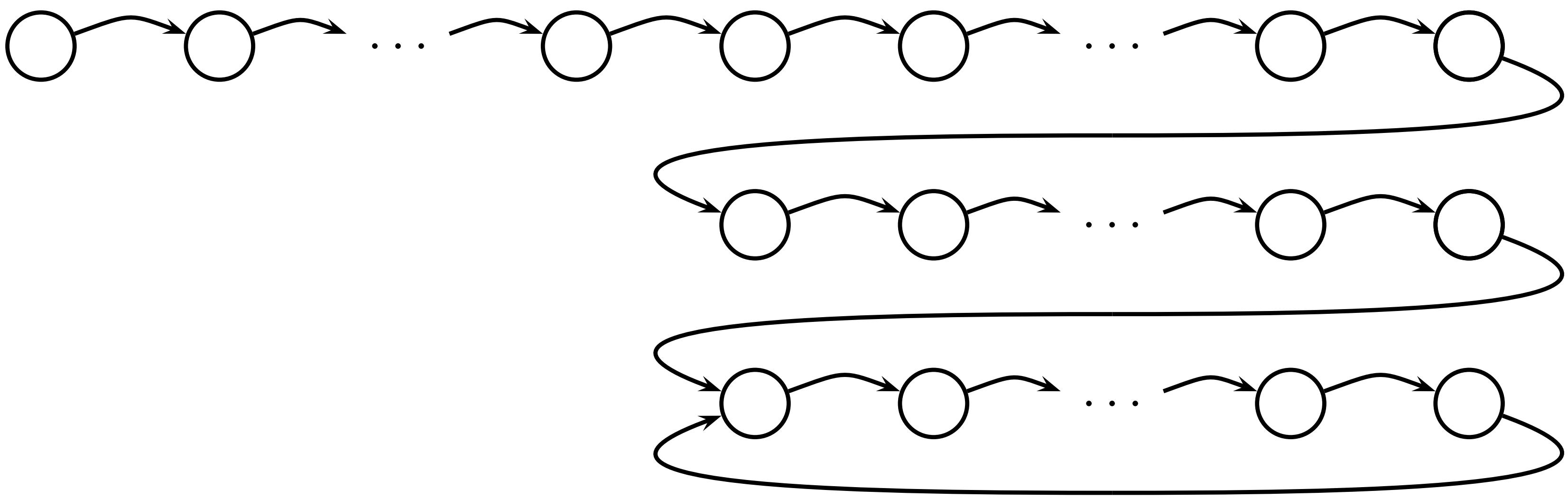


Lemma: For any loop L of type (m, p), for any $NLTL$ formula ϕ with at most $h(\phi)$ nested past-time modalities, and for any $k \geq m + p \cdot h(\phi)$, we have

$$\pi, k \models \phi \qquad \Leftrightarrow \qquad \pi, k + p \models \phi.$$

Model checking a path

A *loop of type* (m, p) is an ultimately periodic KS where the initial part has length m and the periodic part has length p.

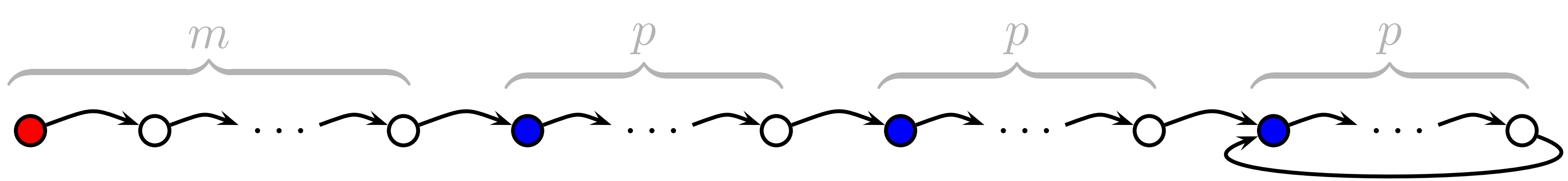


Lemma: For any loop L of type (m, p), for any $NLTL$ formula ϕ with at most $h(\phi)$ nested past-time modalities, and for any $k \geq m + p \cdot h(\phi)$, we have

$$\pi, k \models \phi \qquad \Leftrightarrow \qquad \pi, k + p \models \phi.$$

Model checking a path

A *loop of type* (m, p) is an ultimately periodic KS where the initial part has length m and the periodic part has length p.

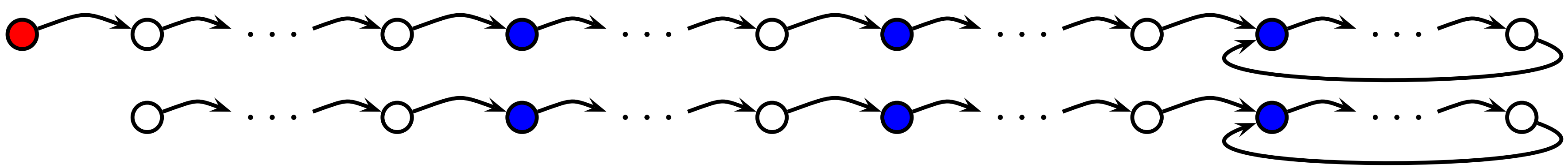


Lemma: For any loop L of type (m, p), for any $NLTL$ formula ϕ with at most $h(\phi)$ nested past-time modalities, and for any $k \geq m + p \cdot h(\phi)$, we have

$$\pi, k \models \phi \qquad \Leftrightarrow \qquad \pi, k + p \models \phi.$$

Model checking a path

A *loop of type* (m, p) is an ultimately periodic KS where the initial part has length m and the periodic part has length p.

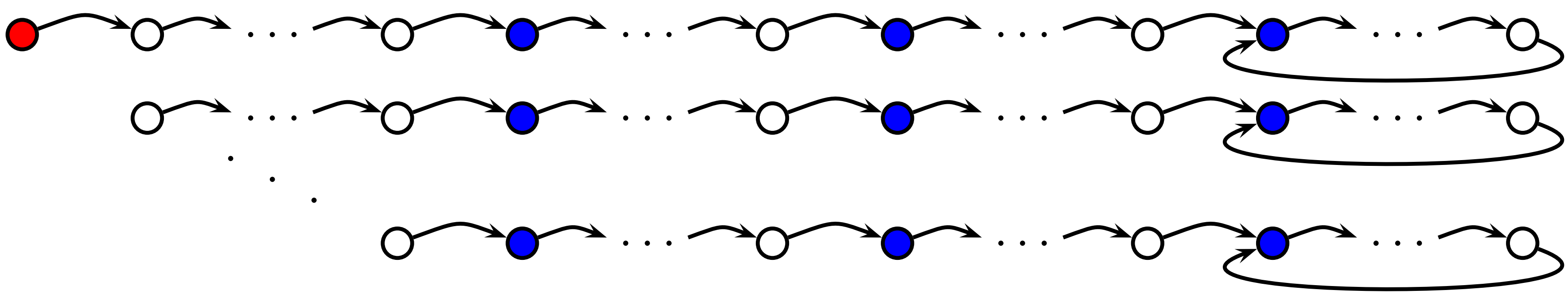


Lemma: For any loop L of type (m, p), for any $NLTL$ formula ϕ with at most $h(\phi)$ nested past-time modalities, and for any $k \geq m + p \cdot h(\phi)$, we have

$$\pi, k \models \phi \qquad \Leftrightarrow \qquad \pi, k + p \models \phi.$$

Model checking a path

A *loop of type* (m, p) is an ultimately periodic KS where the initial part has length m and the periodic part has length p.

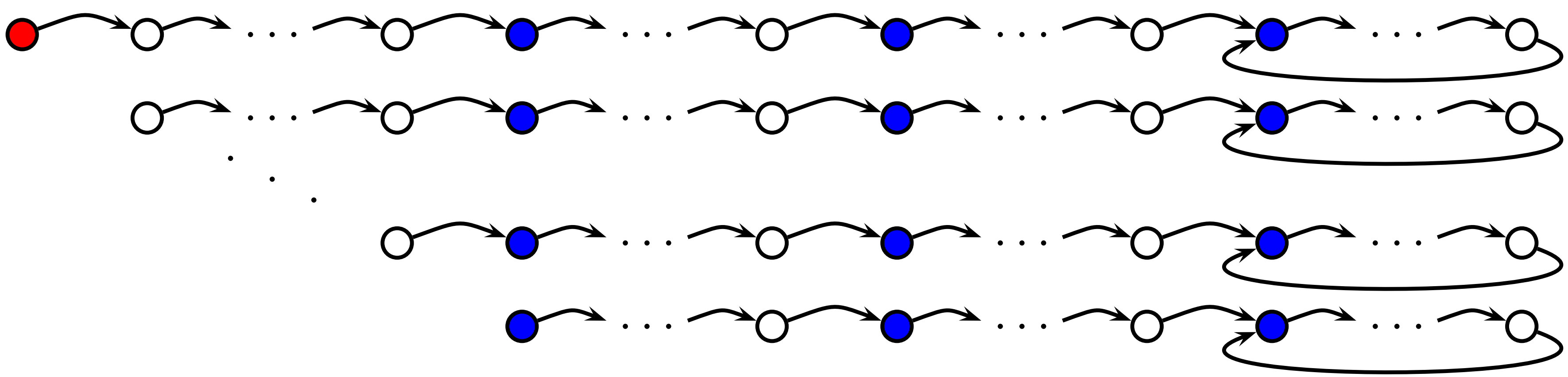


Lemma: For any loop L of type (m, p), for any $NLTL$ formula ϕ with at most $h(\phi)$ nested past-time modalities, and for any $k \geq m + p \cdot h(\phi)$, we have

$$\pi, k \models \phi \qquad \Leftrightarrow \qquad \pi, k + p \models \phi.$$

Model checking a path

A *loop of type* (m, p) is an ultimately periodic KS where the initial part has length m and the periodic part has length p.

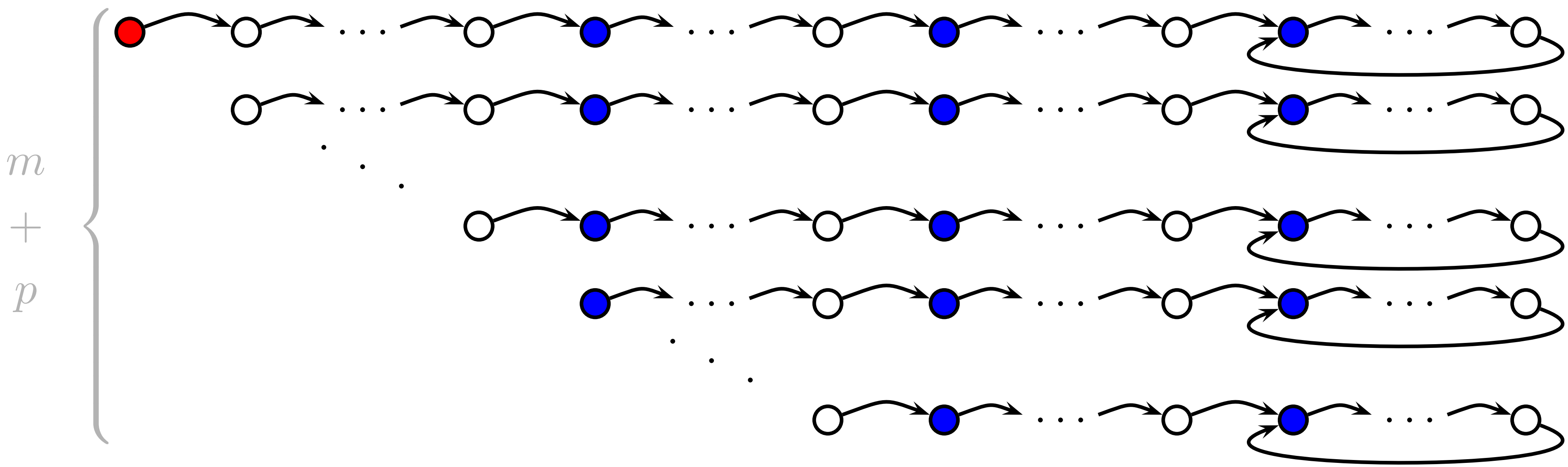


Lemma: For any loop L of type (m, p), for any $NLTL$ formula ϕ with at most $h(\phi)$ nested past-time modalities, and for any $k \geq m + p \cdot h(\phi)$, we have

$$\pi, k \models \phi \qquad \Leftrightarrow \qquad \pi, k + p \models \phi.$$

Model checking a path

A *loop of type* (m, p) is an ultimately periodic KS where the initial part has length m and the periodic part has length p.



Lemma: For any loop L of type (m, p), for any *NLTL* formula ϕ with at most $h(\phi)$ nested past-time modalities, and for any $k \geq m + p \cdot h(\phi)$, we have

$$\pi, k \models \phi \qquad \Leftrightarrow \qquad \pi, k + p \models \phi.$$

Model checking a path

Proposition Model checking *NLTL* along one path is PTIME-hard.

Proof: Reduction from CIRCUIT-VALUE.

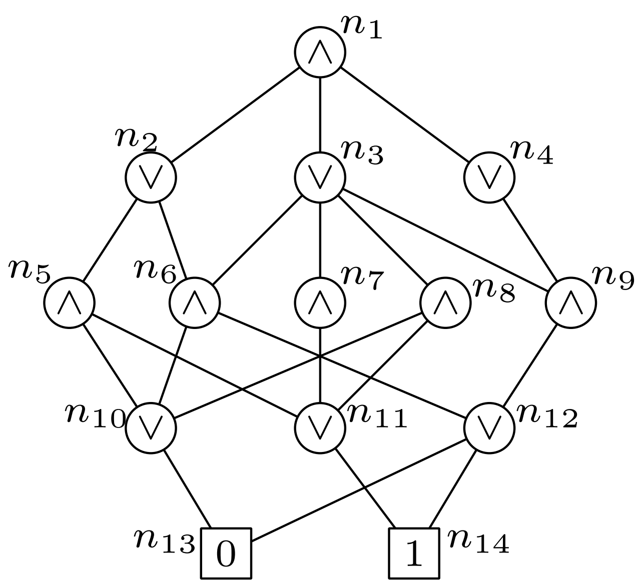
level 4:

level 3:

level 2:

level 1:

level 0:



Model checking a path

Proposition Model checking *NLTL* along one path is PTIME-hard.

Proof: Reduction from CIRCUIT-VALUE.

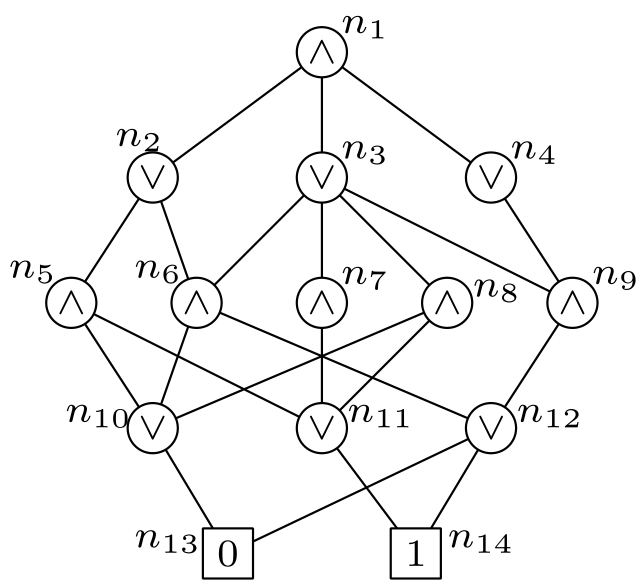
level 4:

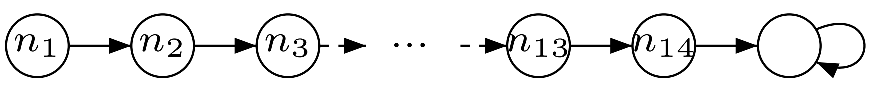
level 3:

level 2:

level 1:

level 0:





Model checking a path

Proposition Model checking *NLTL* along one path is PTIME-hard.

Proof: Reduction from CIRCUIT-VALUE.

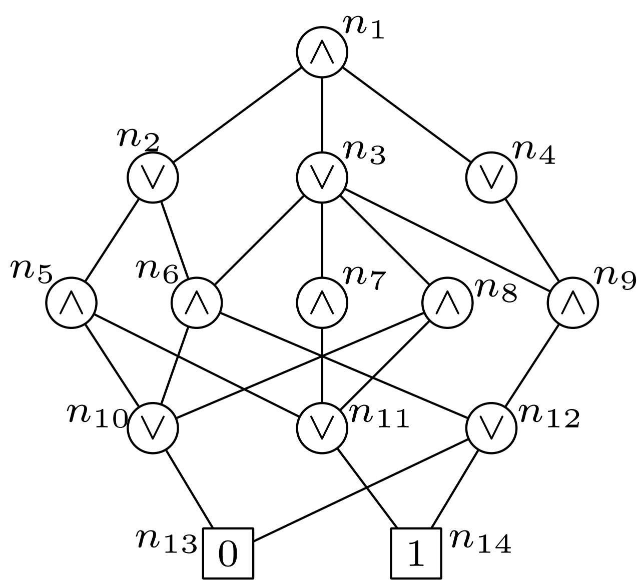
level 4:

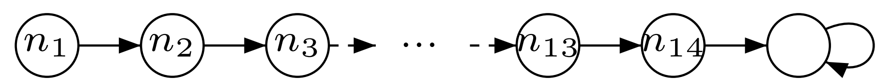
level 3:

level 2:

level 1:

level 0:





$$\varphi_0 \overset{\text{def}}{=} 1$$

Model checking a path

Proposition Model checking *NLTL* along one path is PTIME-hard.

Proof: Reduction from CIRCUIT-VALUE.

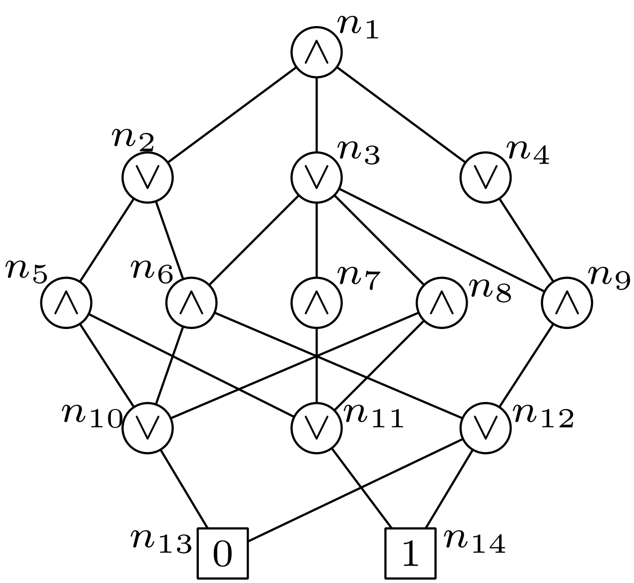
level 4:

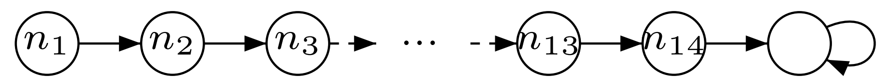
level 3:

level 2:

level 1:

level 0:





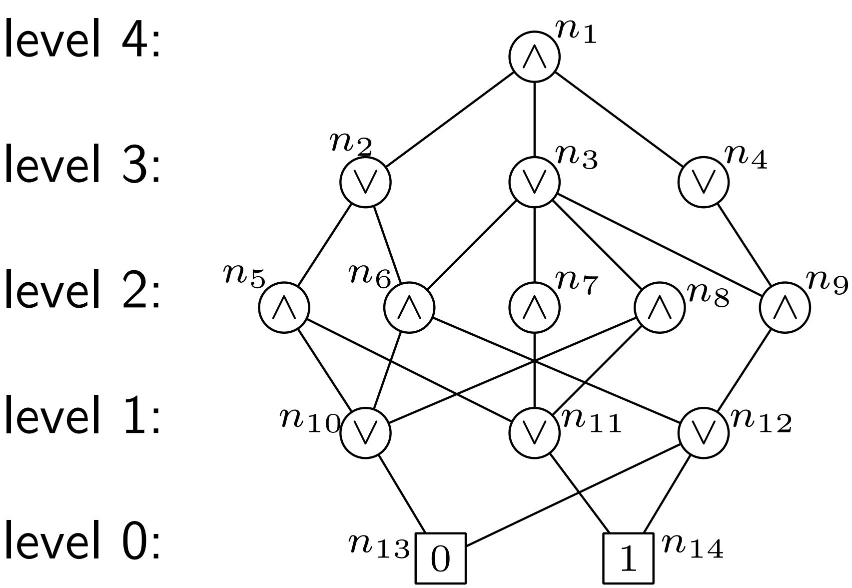
$$\varphi_0 \stackrel{\text{def}}{=} 1$$

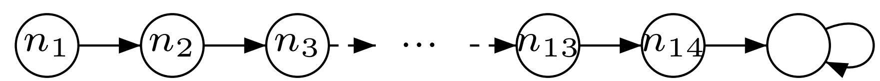
$$\varphi_{2k+1} \stackrel{\text{def}}{=} \mathsf{NF}(\phi_{next} \wedge \varphi_{2k})$$

Model checking a path

Proposition Model checking *NLTL* along one path is PTIME-hard.

Proof: Reduction from CIRCUIT-VALUE.





$$\varphi_0 \stackrel{\text{def}}{=} 1$$

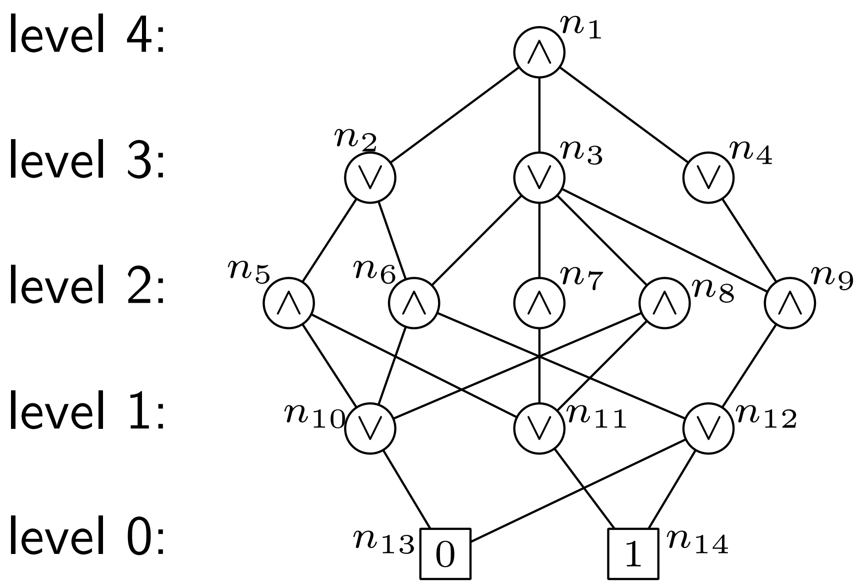
$$\varphi_{2k+1} \stackrel{\text{def}}{=} \mathsf{NF}(\phi_{next} \wedge \varphi_{2k})$$

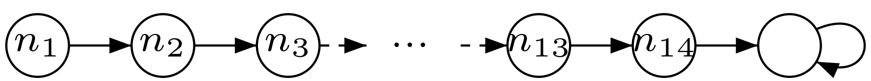
$$\varphi_{2k+2} \stackrel{\text{def}}{=} \mathsf{NG}(\phi_{next} \Rightarrow \varphi_{2k+1})$$

Model checking a path

Proposition Model checking *NLTL* along one path is PTIME-hard.

Proof: Reduction from Circuit-Value.





$$\varphi_0 \stackrel{\text{def}}{=} 1$$

$$\varphi_{2k+1} \stackrel{\text{def}}{=} \mathsf{NF}(\phi_{next} \wedge \varphi_{2k})$$

$$\varphi_{2k+2} \stackrel{\text{def}}{=} \mathsf{NG}(\phi_{next} \Rightarrow \varphi_{2k+1})$$

where $\phi_{next} \stackrel{\text{def}}{=} \bigvee_{(n,n')\in E} (n' \wedge \mathsf{F}^{-1}(n \wedge \neg \mathsf{X}^{-1}\top))$

Conclusion

- $NLTL$ can make specifications easier and more natural.
- That $NLTL$ offers more expressive power can be stated formally as a succinctness gap. The same holds between $PLTL$ and LTL.
- Satisfiability and model checking are EXPSPACE-complete for $NLTL$.

  There is a price for N !
- Model checking a path is PTIME-complete.

Bibliographie

[EVW97] Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. In *LICS'97*, pages 228–235, Warsaw, Poland, 1997. 12th Annual IEEE Symposium on Logic in Computer Science, IEEE Comp. Soc. Press.

[Gab89] Dov M. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In Behnam Banieqbal, Howard Barringer, and Amir Pnueli, editors, *Conference on Temporal Logic in Specification*, volume 398 of *Lect. Notes in Comp. Sci.*, pages 409–448. Springer, 1989.

[Kam68] Hans W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, UCLA, Los Angeles, CA, USA, 1968.

[Var94] Moshe Y. Vardi. Nontraditional applications of automata theory. In Masami Hagiya and John C. Mitchell, editors, *TACS'94*, volume 789 of *Lect. Notes in Comp. Sci.*, pages 575–597, Sendai, Japan, 1994. International Conference on Theoretical Aspects of Computer Software, Springer.

[VW86] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *LICS'86*, pages 332–344, Cambridge, Massachusets, 1986. 1st Annual IEEE Symposium on Logic in Computer Science, IEEE Comp. Soc. Press.

[VW94] Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

Examples of separation

$$\mathsf{G}(\,\mathsf{alarm} \;\Rightarrow\; \mathsf{F}^{-1}\,\mathsf{problem}\,)$$

$$\equiv$$

$$\mathsf{F}^{-1}\mathsf{problem} \;\vee\; \neg\Big(\,(\neg\,\mathsf{problem})\,\mathsf{U}\,(\mathsf{alarm} \;\wedge\; \neg\,\mathsf{problem}\,)\,\Big)$$

$$\equiv_i$$

$$\neg\Big(\,(\neg\,\mathsf{problem})\,\mathsf{U}\,(\mathsf{alarm} \;\wedge\; \neg\,\mathsf{problem}\,)\,\Big)$$

And:

$$\mathsf{G}\,\Big(\,\mathsf{reset} \;\Rightarrow\; \mathsf{N}\,\mathsf{G}\,(\,\mathsf{alarm} \;\Rightarrow\; \mathsf{F}^{-1}\,\mathsf{problem}\,)\Big)$$

$$\equiv_i$$

$$\mathsf{G}\Big[\mathsf{reset} \;\Rightarrow\; \neg\Big(\,(\neg\,\mathsf{problem})\,\mathsf{U}\,(\mathsf{alarm} \;\wedge\; \neg\,\mathsf{problem}\,)\,\Big)$$

Proof of the result of [EVW97]

Claim: Let ϕ_n be a $PLTL$ formula expressing the following property:

"any two future positions that agree on $p_1, \ldots, p_n$ also agree on p_0."

We denote $L_n = \{u \in \{p_0, \ldots, p_n\} \mid u \models \phi_n\}$. From [VW86], we know that L_n is recognized by a Generalized Büchi Automaton $\mathcal{B}$ with $2^{O(|\phi_n|)}$ states.

Let $\{a_0, \ldots, a_{2^n-1}\}$ be a sequence containing all subsets of $\{p_1, \ldots, p_n\}$.

For $K \subseteq \{0, \ldots, 2^n - 1\}$, $w_K = b_0 \cdots b_{2^n-1}$ with $\begin{cases} b_i = a_i & \text{if } i \in K \\ b_i = a_i \cup \{p_0\} & \text{otherwise} \end{cases}$

There are 2^{2^n} such words.

Assume $K \neq K'$. Then $w_K^\omega \models \phi_n$ and $w_{K'} w_K^\omega \not\models \phi_n$. The executions of $\mathcal{B}$ on w_K and $w_{K'}$ cannot lead to the same state. The automaton thus needs at least 2^{2^n} states...