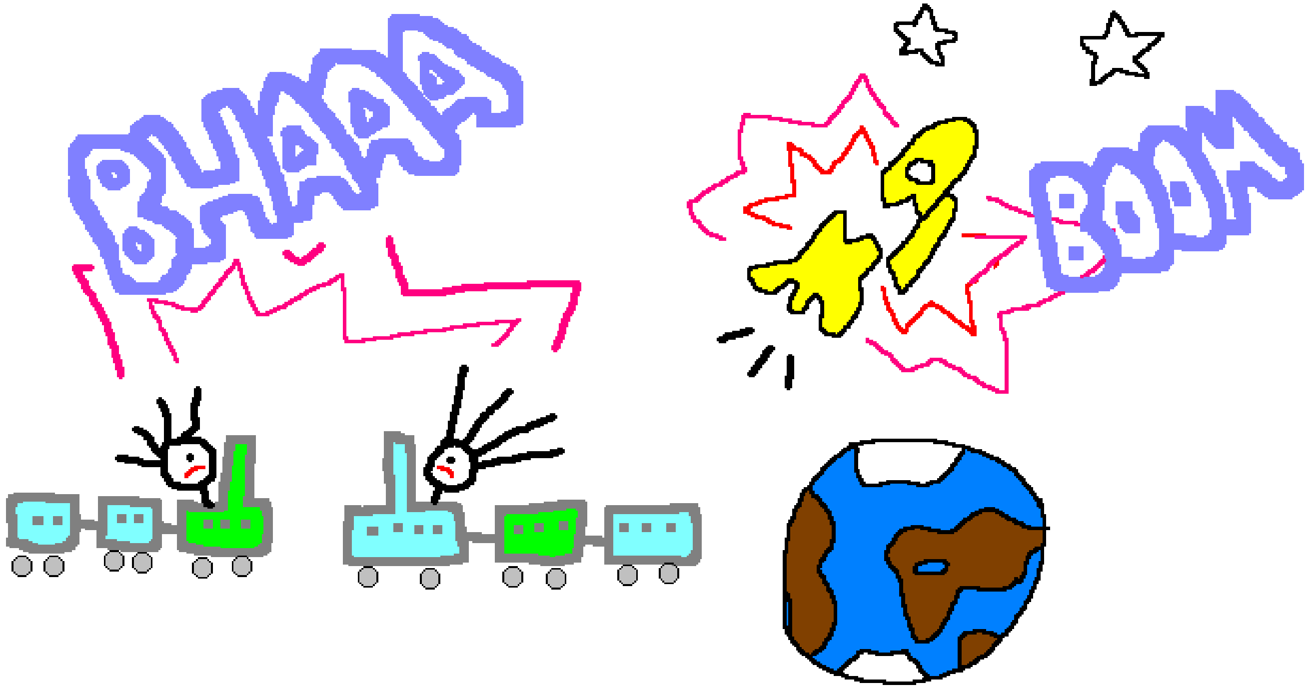
Enriched Modal Logics

Aniello Murano

Università degli Studi di Napoli "Federico II"

Wien – November 7, 2013

Motivations



Is the system correct?

Motivations

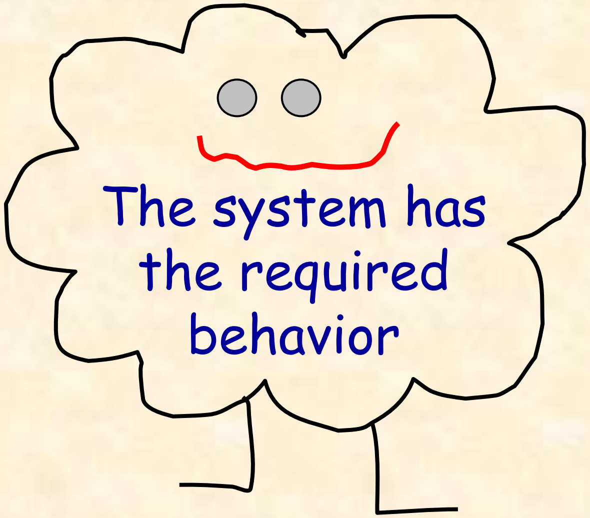
Formal Verification:

- System → A mathematical model M
- Desired Behavior → A formal specification ψ
- Correctness → A formal technique

Motivations

Formal Verification:

- System → A mathematical model M
- Desired Behavior → A formal specification ψ
- Correctness → A formal technique

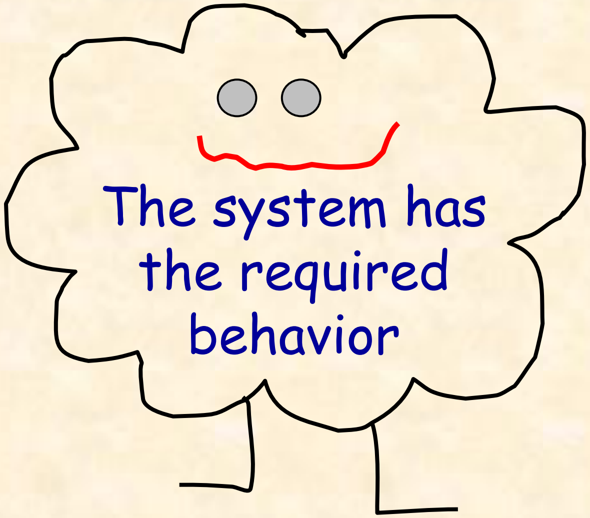


- Model Checking: Does M satisfies ψ ?

Motivations

Formal Verification:

- System → A mathematical model M
- Desired Behavior → A formal specification ψ
- Correctness → A formal technique



- Model Checking: Does M satisfies ψ ?
- Satisfiability: Is there M for ψ ?

A Basic Model: Kripke Structure

- A system can be represented as a **Kripke Structure**: a labeled-state transition graph

$M = (AP, S, S_0, R, Lab)$

  - AP is a set of atomic propositions.
  - S is a finite set of states.
  - $S_0 \subseteq S$ is the set of initial states.
  - $R \subseteq S \times S$ is a transition relation, total: $\forall s \in S, \exists s' . R(s, s')$.
  - $Lab : S \rightarrow 2^{AP}$ labels states with propositions true in that states.

- A path is a system run!

System Specification

- Modal and Temporal logic allow description of the temporal ordering of events

System Specification

- Modal and Temporal logic allow description of the temporal ordering of events
- Two main families of logics:
- Linear-Time Logics (**LTL**)
  - Each moment in time has a unique possible future.
  - LTL expresses path properties based on the paths state labels.
  - Useful for hardware specification.
- Branching-Time Logics (**CTL**, **CTL***, and μ-**CALCULUS**)
  - Each moment in time may split into various possible future.
  - CTL* expresses state properties from which LTL-like properties are satisfied in an existential or universal way .
  - Useful for software specification.

μ-calculus is a very expressive logic

- Can express several practical properties.
- Corresponds to alternating parity tree automata
- Important connections with MSO
- Strictly subsumes classical logics such as CTL, LTL, CTL*, …
- Identifies powerful classes of Description Logics

- Decision problems:
  - Model checking: UP ∩ co-UP
  - Satisfiability: ExpTime-complete

μ-calculus limitations

- Several important constructs cannot be easily translated to the μ-calculus:
  - Inverse Programs to travel relations in backward
  - Graded modalities to enable statements on a number of successors
  - Nominals as propositional variables true exactly in one state

μ-calculus limitations

- Several important constructs cannot be easily translated to the μ-calculus:
  - Inverse Programs to travel relations in backward
  - Graded modalities to enable statements on a number of successors
  - Nominals as propositional variables true exactly in one state

- Extensions of the μ-calculus with these abilities induces families of enriched μ-calculi.
- Similarly, we can define families of enriched temporal logics.

Outline of the talk
I part

- ✓ Motivations
- ❑ Fully enriched μ-calculus

Outline of the talk
I part

- ✓ Motivations
- ❑ Fully enriched μ-calculus
- ❑ Families of enriched μ-calculi
  - ◆ full graded μ-calculus (with inverse programs and graded mod.)
  - ◆ hybrid graded μ-calculus (with graded modalities and nominals)
  - ◆ full hybrid μ-calculus (with inverse programs and nominals)

Outline of the talk
I part

- ✓ Motivations
- ❑ Fully enriched μ-calculus
- ❑ Families of enriched μ-calculi
  - ◆ full graded μ-calculus (with inverse programs and graded mod.)
  - ◆ hybrid graded μ-calculus (with graded modalities and nominals)
  - ◆ full hybrid μ-calculus (with inverse programs and nominals)

- ❑ Satisfiability of fully enriched μ-calculus: Undecidable

Outline of the talk
I part

- ✓ Motivations
- ❑ Fully enriched μ-calculus
- ❑ Families of enriched μ-calculi
  - ◆ full graded μ-calculus (with inverse programs and graded mod.)
  - ◆ hybrid graded μ-calculus (with graded modalities and nominals)
  - ◆ full hybrid μ-calculus (with inverse programs and nominals)

- ❑ Satisfiability of fully enriched μ-calculus: Undecidable
- ❑ Satisfiability of the other families we consider: ExpTime-complete
  - ◆ Upper bound via Fully Enriched Automata (FEA).
  - ◆ The upper bound holds also in case numbers are coded in binary

Outline of the talk
II part

- Graded Computation Tree Logic (GCTL)
- ExpTime solution of the satisfiability problem for graded numbers coded in unary/binary

Outline of the talk
II part

- Graded Computation Tree Logic (GCTL)
- ExpTime solution of the satisfiability problem for graded numbers coded in unary/binary
- Open questions on GCTL and its extensions:
  - GCTL*, PGCTL/PGCTL*, etc..

Outline of the talk
II part

- Graded Computation Tree Logic (GCTL)

- ExpTime solution of the satisfiability problem for graded numbers coded in unary/binary

- Open questions on GCTL and its extensions:
  - GCTL*, PGCTL/PGCTL*, etc..

- Some achievements in open system verification.

I part: Enriched µ-calculi

Some known results

- Satisfiability for Fully enriched μ-calculus is undecidable [Bonatti, Peron 2004]

Some known results

- Satisfiability for Fully enriched μ-calculus is undecidable [Bonatti, Peron 2004]
- ExpTime-completeness of satisfiability for enriched μ-calculi:
  - μ-calculus with inverse programs [Vardi'98]
  - μ-calculus with graded modalities [Kupferman,Sattler,Vardi'02]
  - full hybrid logic [Sattler,Vardi'01]
  - full graded logic in unary coding [Calvanese, De Giacomo, Lenzerini'01]

The fully enriched μ-calculus

- The μ-calculus is a propositional modal logic with least(μ) and greatest (ν) fixpoint operators [Kozen 1983].

- The fully enriched μ-calculus extends the μ-calculus with
  - graded modalities: $\langle$**n**,$\alpha\rangle$ (atleast formulas) and **[n**,α**]** (allbut formulas)
  - nominals propositions: Nominal set **Nom**
  - inverse programs: Use of both program sets **Prog** and **Prog-**

The fully enriched μ-calculus (Syntax)

- Let AP, Var, Prog, and Nom be sets of atomic proposition, propositional variables, atomic, programs and nominals
- Syntax:

  $\varphi := true \mid false \mid p \mid \neg p \mid y \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle n,\alpha\rangle\varphi \mid [n,\alpha]\varphi \mid \mu y.\ \varphi(y) \mid \nu y.\ \varphi(y)$

  where $p \in AP \cup Nom$, $y \in Var$, $n \in \mathbb{N}$, and α is a program or its converse

The fully enriched μ-calculus (Syntax)

- Let AP, Var, Prog, and Nom be sets of atomic proposition, propositional variables, atomic, programs and nominals
- Syntax:

  $\varphi := true \mid false \mid p \mid \neg p \mid y \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle n,\alpha\rangle\varphi \mid [n,\alpha]\varphi \mid \mu y.\, \varphi(y) \mid \nu y.\, \varphi(y)$

  where $p \in AP \cup Nom$, $y \in Var$, $n \in N$, and α is a program or its converse

- Fragments of the fully enriched μ-calculus:
  - full graded μ-calculus (without nominals)
  - hybrid graded μ-calculus (without inverse programs)
  - full hybrid μ-calculus (without graded modalities)

Semantics: The enriched model

- The semantics of the fully enriched μ-calculus is given with respect to enriched Kripke structures

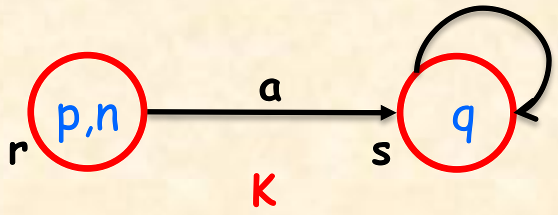
$$K = (AP \cup Nom, W, W_0, R, Lab)$$

Semantics: The enriched model

- The semantics of the fully enriched μ-calculus is given with respect to enriched Kripke structures

$$K = (AP \cup Nom, W, W_0, R, Lab)$$

- In particular, R and Lab are enriched as follows:
  - $R : Prog \rightarrow 2^{W \times W}$ assigns to programs transitions relation over S
  - Lab: $AP \cup Nom \rightarrow 2^W$ assigns to propositions and nominal sets of states, where those assigned to each nominal are singletons.



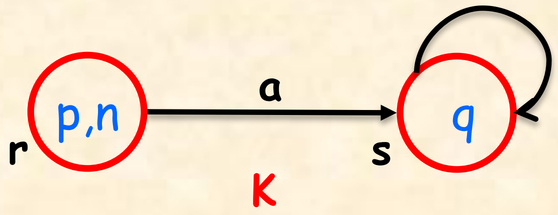
Semantics: The enriched model

- The semantics of the fully enriched μ-calculus is given with respect to enriched Kripke structures

$$K = (AP \cup Nom,\ W,\ W_0,\ R,\ Lab)$$

- In particular, R and Lab are enriched as follows:
  - $R : Prog \rightarrow 2^{W \times W}$ assigns to programs transitions relation over S
  - $Lab: AP \cup Nom \rightarrow 2^W$ assigns to propositions and nominal sets of states, where those assigned to each nominal are singletons.

- Given a Kripke structure, atomic propositions and boolean connectivities are interpreted as usual:
  - K satisfies the nominal n at the starting state r, since Lab(n)={s}
  - K does not satisfy q at r, but at s.

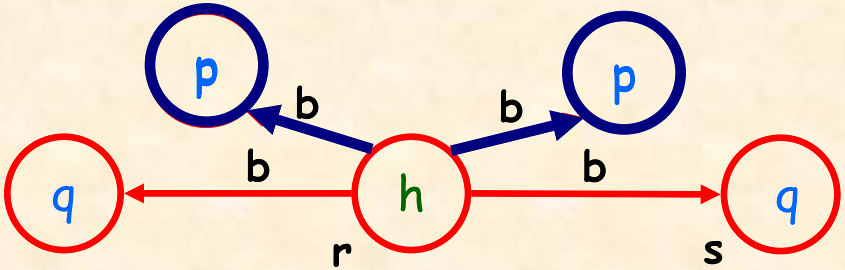


Semantics

- For a Kripke structure, the new modalities are interpreted as follows.
- $\langle n,\alpha\rangle\varphi$ holds in w if φ holds at least in n+1 α-successors of w.
- $[n,\alpha]\varphi$ holds in w if φ holds in all but at most n α-successors of w.

Semantics

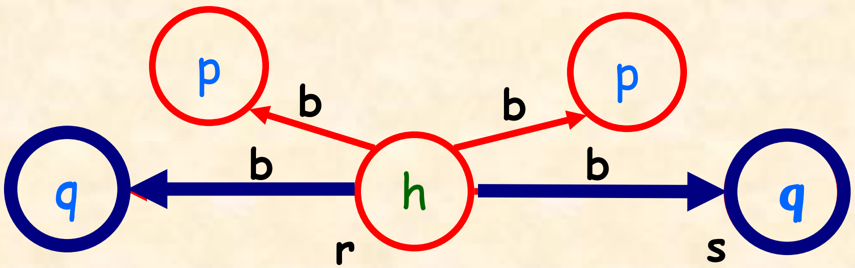
- For a Kripke structure, the new modalities are interpreted as follows.
- $\langle n,\alpha\rangle\varphi$ holds in w if φ holds at least in n+1 α-successors of w.
- $[n,\alpha]\varphi$ holds in w if φ holds in all but at most n α-successors of w.



- In r, $\langle 1,b\rangle$p holds

Semantics

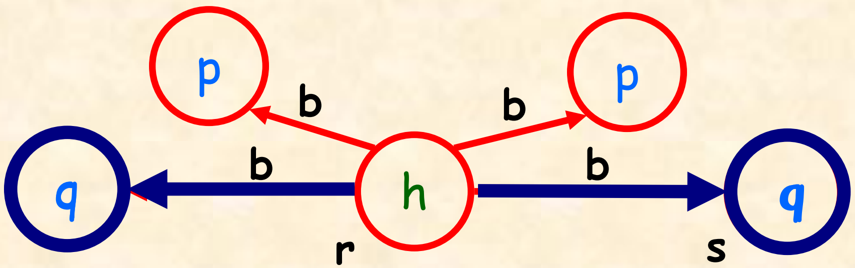
- For a Kripke structure, the new modalities are interpreted as follows.
- $\langle n,\alpha\rangle\varphi$ holds in w if φ holds at least in n+1 α-successors of w.
- $[n,\alpha]\varphi$ holds in w if φ holds in all but at most n α-successors of w.



- In r, $\langle 1,b\rangle p$ holds
- In r, $[1,b]p$ does not hold.

Semantics

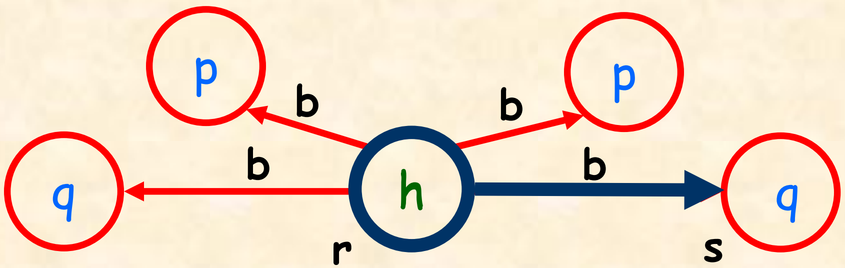
- For a Kripke structure, the new modalities are interpreted as follows.
- $\langle n,\alpha\rangle\varphi$ holds in w if φ holds at least in n+1 α-successors of w.
- $[n,\alpha]\varphi$ holds in w if φ holds in all but at most n α-successors of w.



- In r, $\langle 1,b\rangle p$ holds
- In r, $[1,b]p$ does not hold.
- In r, $[2,b]p$ holds.

Semantics

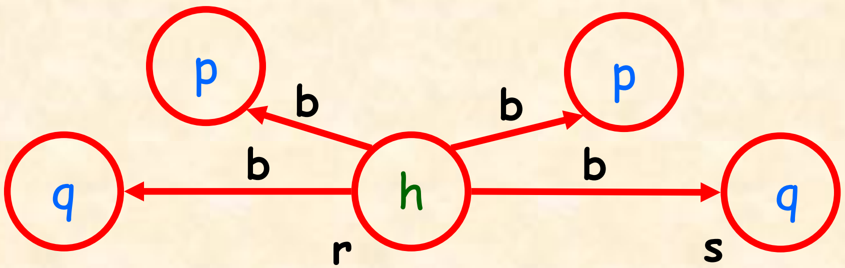
- For a Kripke structure, the new modalities are interpreted as follows.
- $\langle n,\alpha\rangle\varphi$ holds in w if φ holds at least in n+1 α-successors of w.
- $[n,\alpha]\varphi$ holds in w if φ holds in all but at most n α-successors of w.



- In r, $\langle 1,b\rangle p$ holds
- In r, $[1,b]p$ does not hold.
- In r, $[2,b]p$ holds.
- In s, $\langle 0,b^{-}\rangle h$ holds

Semantics

- For a Kripke structure, the new modalities are interpreted as follows.
- $\langle n,\alpha\rangle\varphi$ holds in w if φ holds at least in n+1 α-successors of w.
- $[n,\alpha]\varphi$ holds in w if φ holds in all but at most n α-successors of w.



- In r, $\langle 1,b\rangle p$ holds
- In r, $[1,b]p$ does not hold.
- In r, $[2,b]p$ holds.
- In s, $\langle 0,b^{-}\rangle h$ holds

- ν and μ are useful to express liveness and safety:
  - AGp: p always true along all a-paths is $\nu X.\ p \wedge [0,a]X$
  - EFp: there exists an a-path where p eventually holds is $\mu X.\ p \vee \langle 0,a\rangle X$

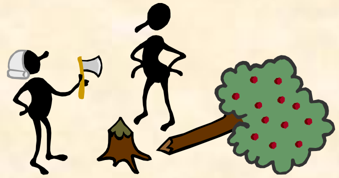
- Note that $\langle 0,\alpha\rangle\varphi$ is $\langle\alpha\rangle\varphi$ and $[0,\alpha]\varphi$ is $[\alpha]\varphi$

Structure properties

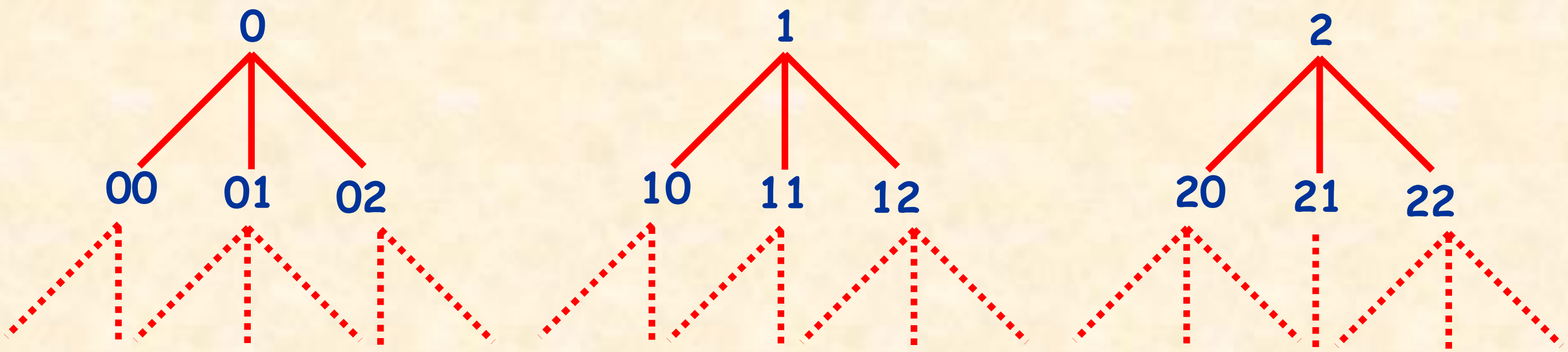
- In branching-time temporal logic, important model features to symplify decisions reasonings are:
- Finite-model property:
  - Is there a finite model satisfying the formula
  - It is possible to use exhaustive (brute-force) methods!
- Tree-model property:
  - Is there a tree-model shape satisfying the formula
  - It is possible to use tree automata !

- In enriched μ-calculus we need **forest structures** as models

Forest structures

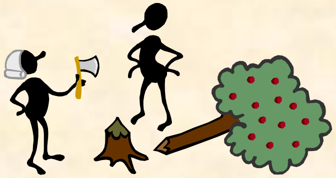


- A forest $F \subseteq N^+$ is a collection of trees:

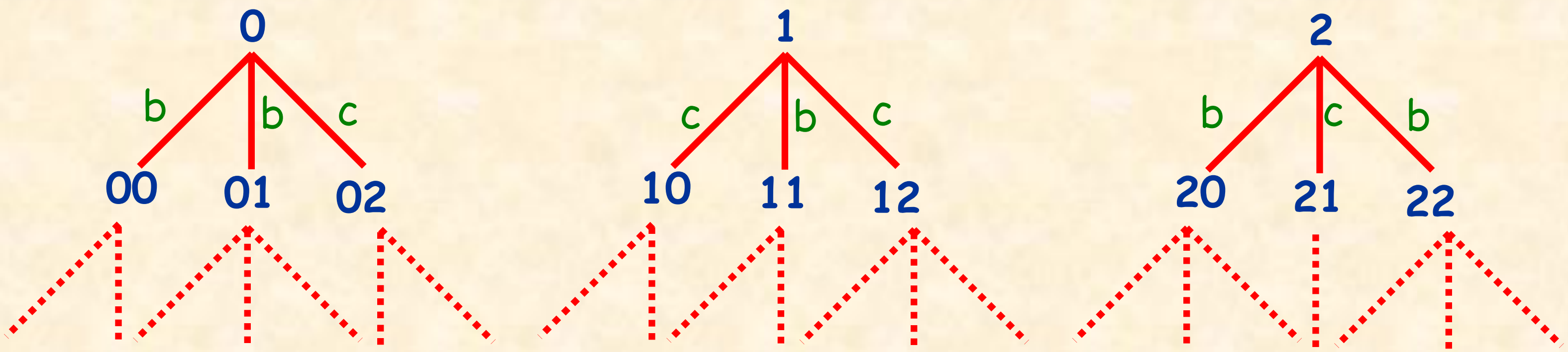


- The elements of F are nodes, the degree of F is the maximum number of node's successors, and 0, 1, and 2 are roots of F.
- The set $T = \{r \cdot x \mid x \in N^* \text{ and } r \cdot x \in F\}$ is the tree of F rooted in r

Forest structures

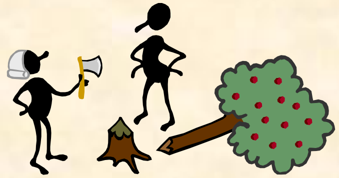


- A forest $F \subseteq N^+$ is a collection of trees:

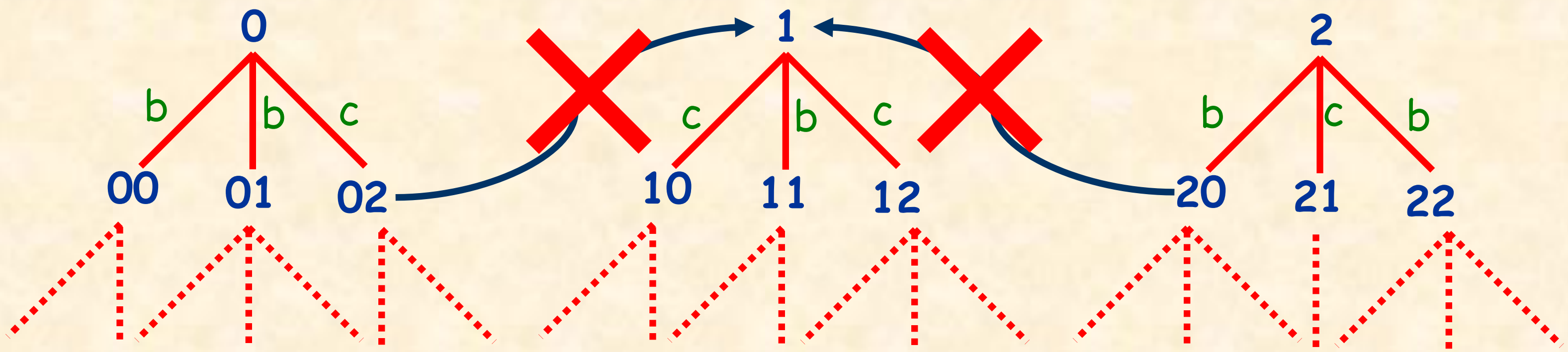


- The elements of F are nodes, the degree of F is the maximum number of node's successors, and 0, 1, and 2 are roots of F.
- The set $T = \{r \cdot x \mid x \in N^* \text{ and } r \cdot x \in F\}$ is the tree of F rooted in r
- A Kripke structure K is a *forest structure* if it induces a forest:
  - Nodes W represent a forest and the relation R is defined over nodes, where each pair of successive nodes is labeled with one atomic program or its converse.

Forest structures

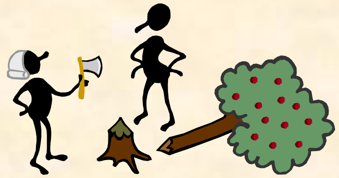


- A forest $F \subseteq N^{+}$ is a collection of trees:

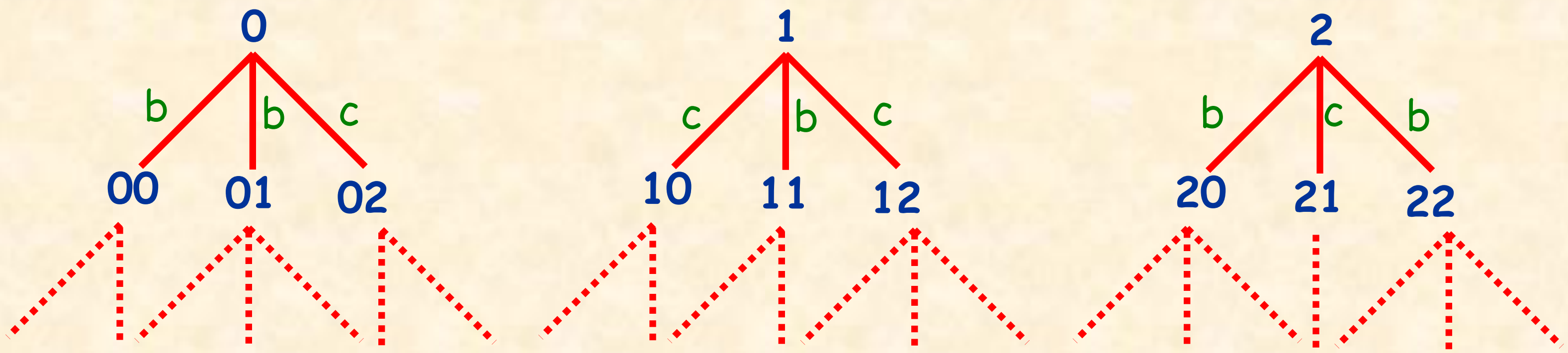


- The elements of F are nodes, the degree of F is the maximum number of node's successors, and 0, 1, and 2 are roots of F.
- The set $T = \{r \cdot x \mid x \in N^{*} \text{ and } r \cdot x \in F\}$ is the tree of F rooted in r
- A Kripke structure K is a *forest structure* if it induces a forest:
  - Nodes W represent a forest and the relation R is defined over nodes, where each pair of successive nodes is labeled with one atomic program or its converse.
- A Kripke structure K is a *quasi forest structure* if it becomes a forest structure after deleting all the edges entering a root of W.

Forest structures

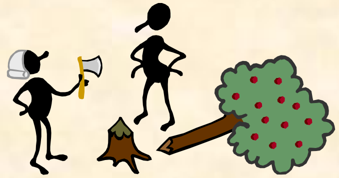


- A forest $F \subseteq N^+$ is a collection of trees:

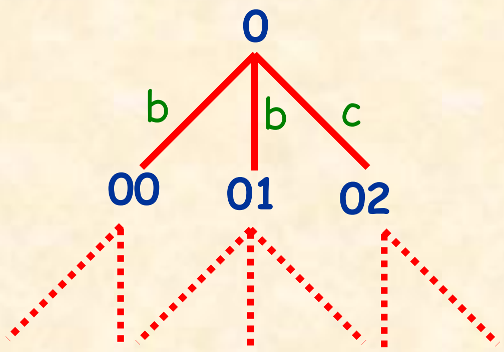


- The elements of F are nodes, the degree of F is the maximum number of node's successors, and 0, 1, and 2 are roots of F.
- The set $T = \{r \cdot x \mid x \in N^* \text{ and } r \cdot x \in F\}$ is the tree of F rooted in r
- A Kripke structure K is a *forest structure* if it induces a forest:
  - Nodes W represent a forest and the relation R is defined over nodes, where each pair of successive nodes is labeled with one atomic program or its converse.
- A Kripke structure K is a *quasi forest structure* if it becomes a forest structure after deleting all the edges entering a root of W.

Forest structures



- A forest F $\subseteq$ N^+ is a collection of trees:



- The elements of F are nodes, the degree of F is the maximum number of node's successors, and 0, 1, and 2 are roots of F.
- The set T= {r·x | x $\in$ N^* and r·x $\in$ F } is the tree of F rooted in r
- A Kripke structure K is a *forest structure* if it induces a forest:
  - Nodes W represent a forest and the relation R is defined over nodes, where each pair of successive nodes is labeled with one atomic program or its converse.
- A Kripke structure K is a *quasi forest structure* if it becomes a forest structure after deleting all the edges entering a root of W.
- K is a *tree structure* if W consists of a single tree.

Forest and tree model property

- Given a sentence φ of the full graded μ-calculus with m atleast subsentences and counting up to b

Forest and tree model property

- Given a sentence φ of the full graded μ-calculus with m atleast subsentences and counting up to b

φ is satisfiable

⇓

φ has a tree model whose degree is at most $m \cdot (b+1)$.

Forest and tree model property

- Given a sentence φ of the full graded μ-calculus with m atleast subsentences and counting up to b

φ is satisfiable

⇓

φ has a tree model whose degree is at most $m \cdot (b+1)$.

- The hybrid graded μ-calculus does not enjoy the tree model property.

Forest and tree model property

- Given a sentence φ of the full graded μ-calculus with m atleast subsentences and counting up to b

φ is satisfiable

⬇

φ has a tree model whose degree is at most $m \cdot (b+1)$.

- The hybrid graded μ-calculus does not enjoy the tree model property.

- Given a sentence φ of the hybrid graded μ-calculus with k nominals, m atleast subsentences and counting up to b

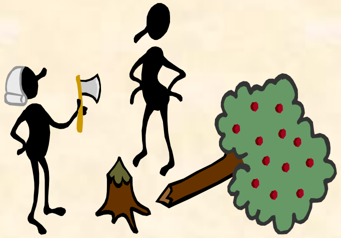
φ is satisfiable

⬇

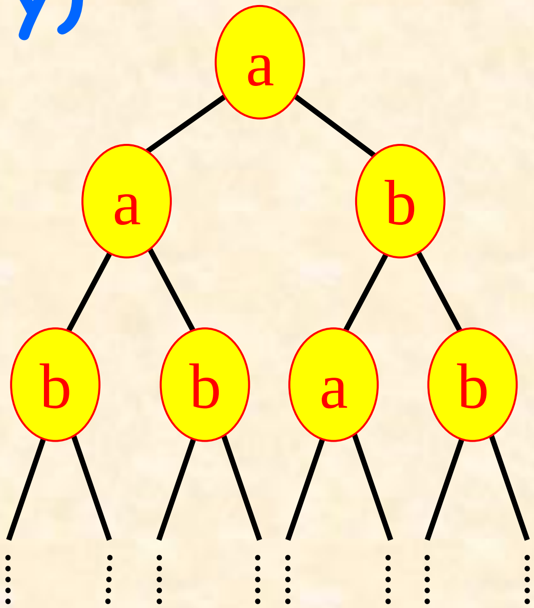
φ has a quasi forest model
whose degree is at most $\max\{k+1, m \cdot (b+1)\}$

Solving enriched mu-calculi

- We use an automata-theoretic approach.
- In modal μ-calulus, we translate a formula to an alternating parity tree automaton anch check for its emptiness.
  - The translation is polynomial
  - Checking for emptiness can be done in ExpTime
  - Satisfiability of μ-calculus is solvable in ExpTime.
- For the enriched μ-calculi, we need an enriched version of parity tree automata.
- Let us first recall alternating automata on infinite tree…



Nondeterministic (binary) tree automata: NTA

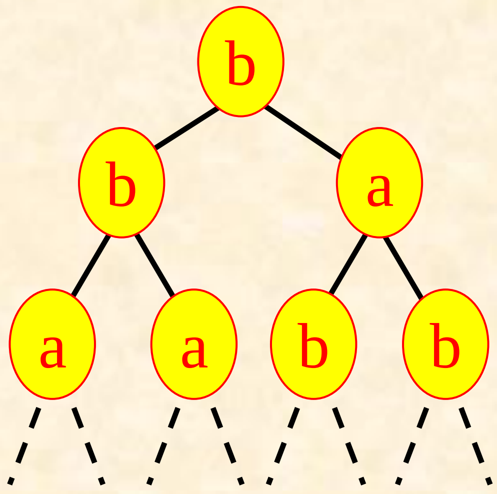


a (Σ-labeled) tree t

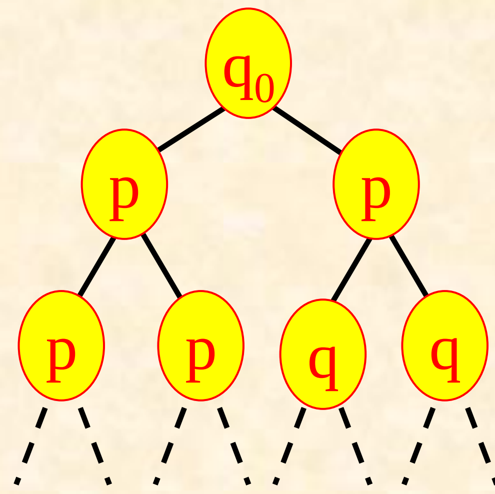
- A infinite (binary) tree is **$t : \{0,1\}^* \rightarrow \Sigma$**
- A **path** is an **infinite** sequence of nodes starting at the root
- An **NTA** is a tuple **A** = < Q, Σ, δ, Q_0, **F** >
  - **δ** : $Q \times \Sigma \rightarrow 2^{Q \times Q}$ is a tree transition relation
  - Runs are binary trees labeled with states accordingly to **δ**
  - **F** is an acceptance condition satisfied on each path of a run

Runs

- A **run** r : $\{0,1\}^* \rightarrow Q$ is built in accordance with δ and $r(\varepsilon) \in Q_0$. Thus, runs are Q-labeled trees.
- Let $(q,q) \in \delta(p,a)$ and q_0 initial state



a tree t

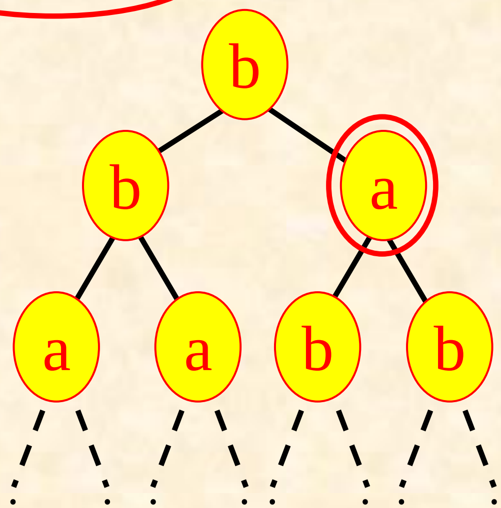


the corresponding run r

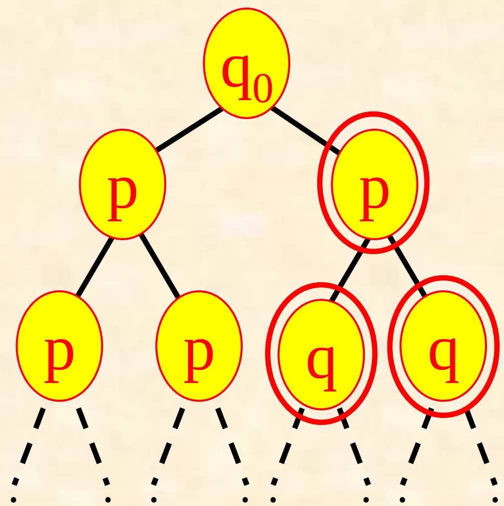
Runs

- A **run** $r : \{0,1\}^* \rightarrow Q$ is built in accordance with δ and $r(\varepsilon) \in Q_0$. Thus, runs are Q-labeled trees.

❿Let $(q,q) \in \delta(p,a)$ and q_0 initial state



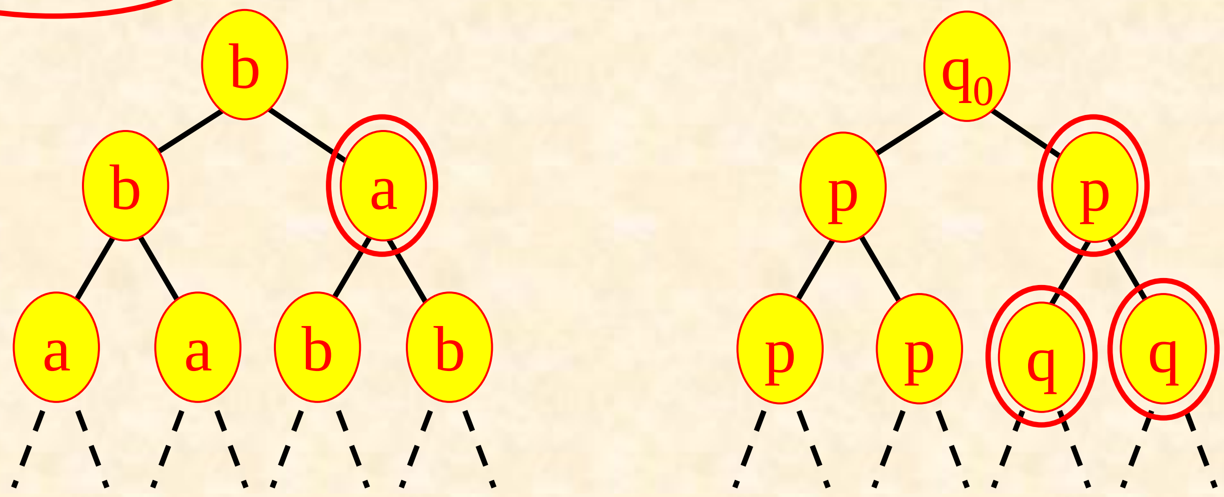
a tree t



the corresponding run r

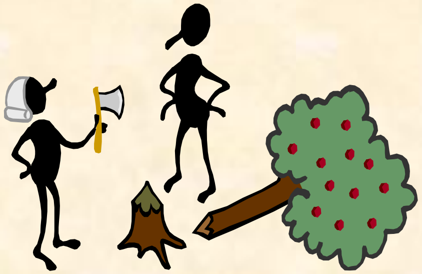
Runs

- A **run** r : $\{0,1\}^* \rightarrow Q$ is built in accordance with δ and $r(\varepsilon) \in Q_0$. Thus, runs are Q-labeled trees.
- Let $(q,q) \in \delta(p,a)$ and q_0 initial state



a tree t the corresponding run r

- A run is **accepting** if the acceptance condition is satisfied on every path



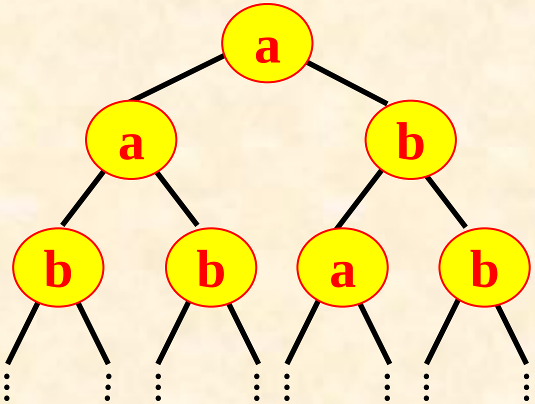
Alternating automata on infinite trees

- An alternating (finite-state) automaton on infinite Σ-labeled D-trees is a tuple

$$A = \langle Q, \Sigma, \delta, q_0, F \rangle$$

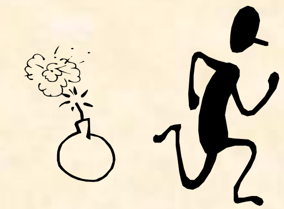
  - $\delta : (Q \times \Sigma) \rightarrow B^+(D \times Q)$
  - positive Boolean formulas of pairs of directions and states

For example
$\delta(p,a)=$
$(1,p) \wedge (1,q)$

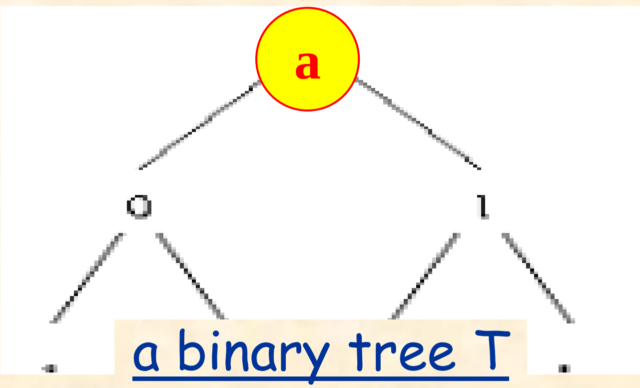


Σ-labeled binary tree

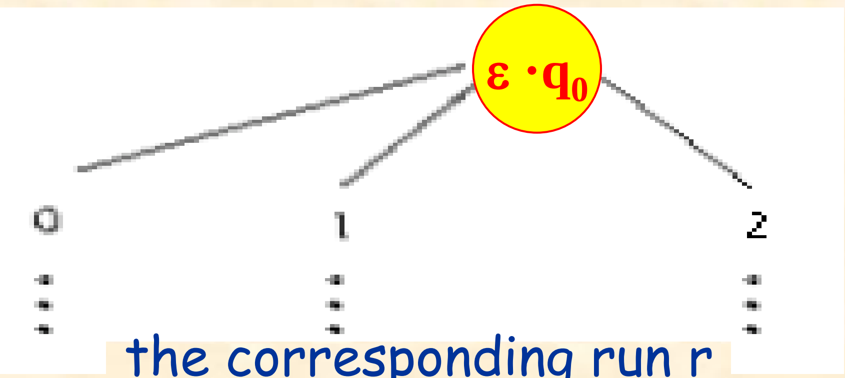
Runs



- A run on a Σ-labeled D-trees is a $(D^* \times Q)$-labeled tree. The root is labeled with (ε, q_0) and labels of each node and its successors must satisfy the δ

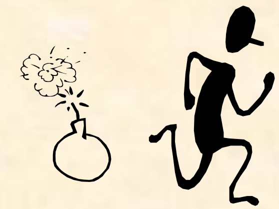


a binary tree T

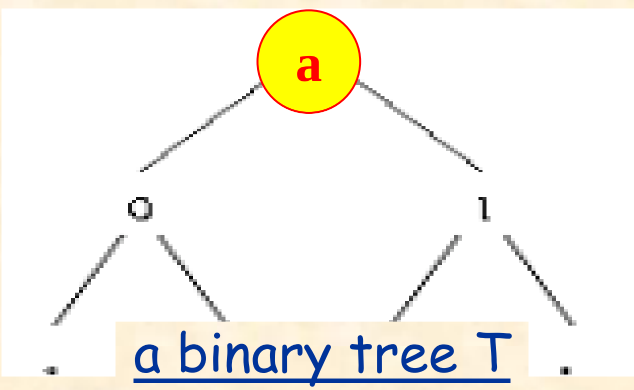


the corresponding run r

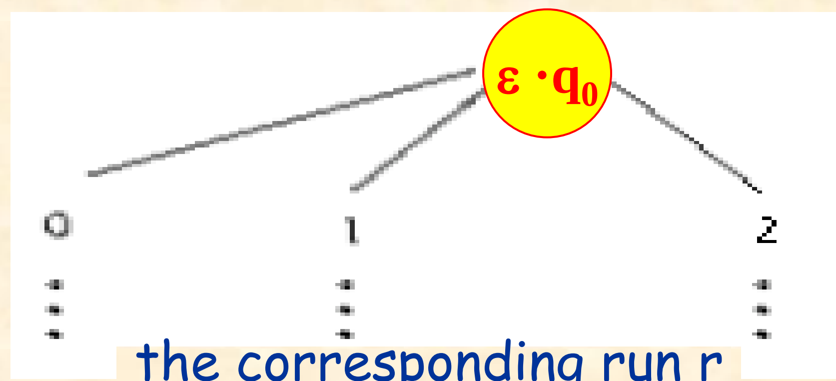
Runs



- A run on a Σ-labeled D-trees is a $(D^* \times Q)$-labeled tree. The root is labeled with (ε, q_0) and labels of each node and its successors must satisfy the δ
- $\delta(q_0,a)=((0,q_1)\vee(0,q_2)) \wedge (0,q_3) \wedge (1,q_3)$
- Let $S= \{(0,q_1), (0,q_3), (1,q_3)\}$.

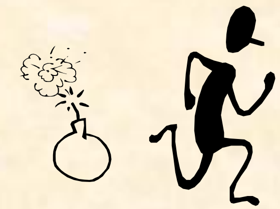


a binary tree T

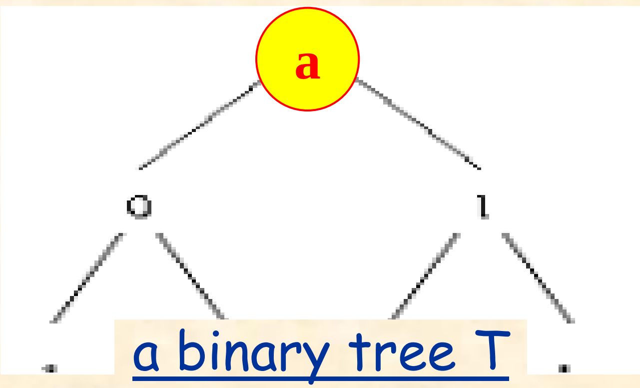


the corresponding run r

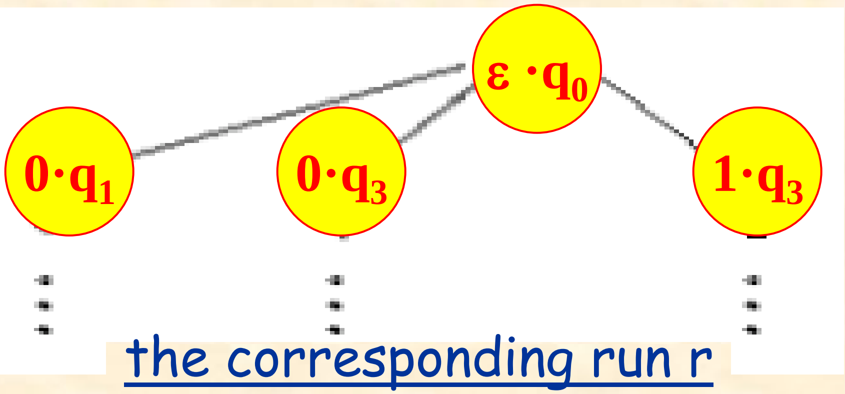
Runs



- A run on a Σ-labeled D-trees is a $(D^* \times Q)$-labeled tree. The root is labeled with (ε, q_0) and labels of each node and its successors must satisfy the δ
- $\delta(q_0,a)=((0,q_1)\vee(0,q_2)) \wedge (0,q_3) \wedge (1,q_3)$
- Let $S= \{(0,q_1), (0,q_3), (1,q_3)\}$.



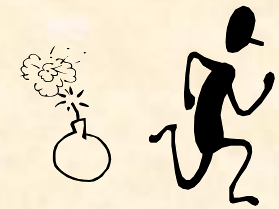
a binary tree T



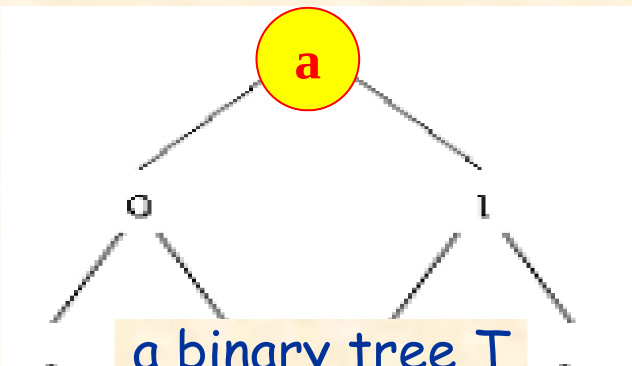
the corresponding run r

- There is no one-to-one correspondence between nodes of T and r

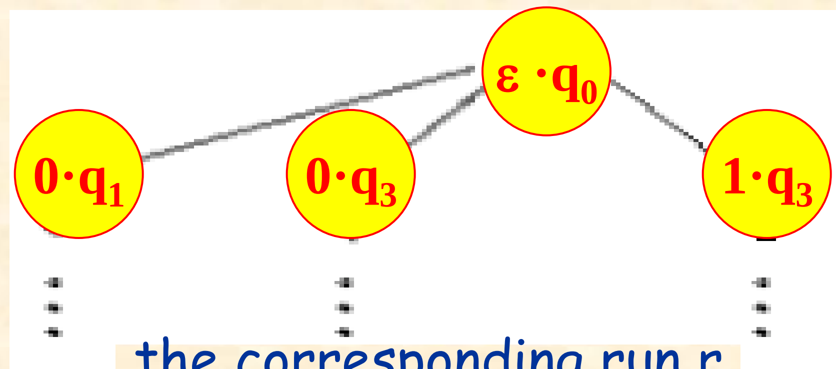
Runs



- A run on a Σ-labeled D-trees is a $(D^* \times Q)$-labeled tree. The root is labeled with (ε, q_0) and labels of each node and its successors must satisfy the δ
- $\delta(q_0,a)=((0,q_1)\vee(0,q_2)) \wedge (0,q_3) \wedge (1,q_3)$
- Let $S= \{(0,q_1), (0,q_3), (1,q_3)\}$.



a binary tree T

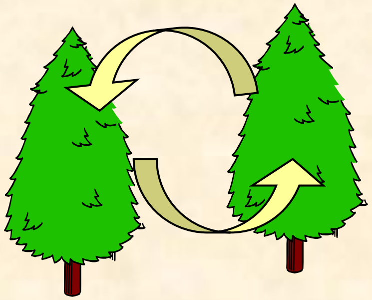


the corresponding run r

- There is no one-to-one correspondence between nodes of T and r
- As in nondeterministic automata, a run is accepting if the acceptance condition is satisfied on every path.

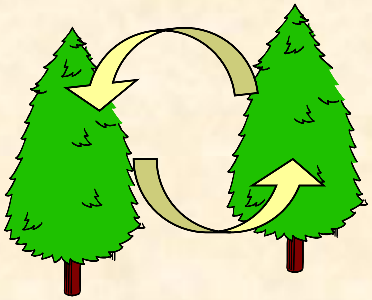
Fully Enriched Automata

- Fully enriched automata (FEA) run on infinite labeled forests $\langle T, V \rangle$.
- FEA generalize alternating automata on infinite trees as the fully enriched μ-calculus extends the standard μ-calculus:



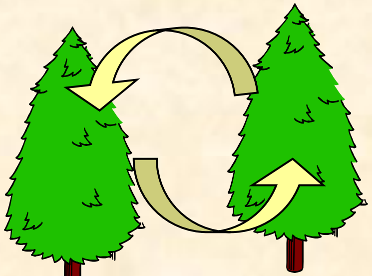
Fully Enriched Automata

- Fully enriched automata (FEA) run on infinite labeled forests $\langle T,V \rangle$.
- FEA generalize alternating automata on infinite trees as the fully enriched μ-calculus extends the standard μ-calculus:
  - Move up to a predecessor of a node (by analogy with inverse programs)
  - Move down to at least n or all but n successors (by analogy with graded modalities)
  - Jump directly to the roots of the input forest (which are the analogues of nominals).



Fully Enriched Automata

- Fully enriched automata (FEA) run on infinite labeled forests $\langle T,V \rangle$.
- FEA generalize alternating automata on infinite trees as the fully enriched μ-calculus extends the standard μ-calculus:
  - Move up to a predecessor of a node (by analogy with inverse programs)
  - Move down to at least n or all but n successors (by analogy with graded modalities)
  - Jump directly to the roots of the input forest (which are the analogues of nominals).

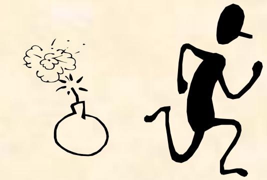


- $\delta(q,\sigma)$ is a positive boolean combination of pairs of directions and states.
- Formally,
  - $\delta : Q \times \Sigma \rightarrow B^+(D_b \times Q)$, where D_b can be -1, ε, $\langle root \rangle$, [root], $\langle n \rangle$, or [n], with $0 \leq n \leq b$.
  - (-1, q) and (ε, q) send a copy to the predecessor and to the current node.
  - ($\langle$root$\rangle$, q) and ([root], q) send a copy to some or all roots of the forest.
  - ($\langle$n$\rangle$, q) and ([n], q) send a copy in state q to n+1 and all but n successors of the current node, respectively.

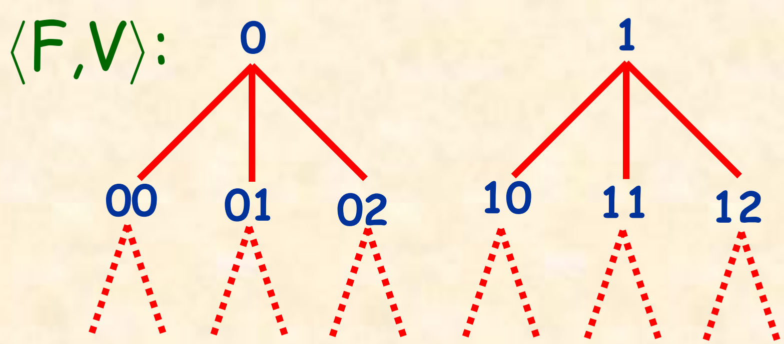
Runs for FEA

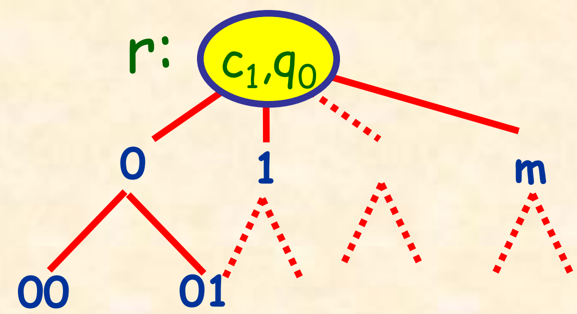
- For a FEA A with a transition δ: $Q \times \Sigma \rightarrow B^{+}(D_b \times Q)$
- A run over a forest $\langle F,V \rangle$ is a $(F \times Q)$-labeled tree, built in accordance with δ and $r(\varepsilon) = (c, q_0)$, for a root c of F.

Runs for FEA



- For a FEA A with a transition δ: $Q \times \Sigma \rightarrow B^+(D_b \times Q)$
- A run over a forest $\langle F,V \rangle$ is a $(F \times Q)$–labeled tree, built in accordance with δ and $r(\varepsilon) = (c, q_0)$, for a root c of F.

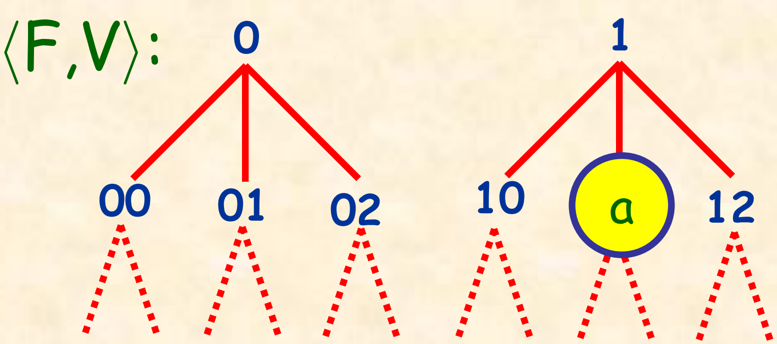


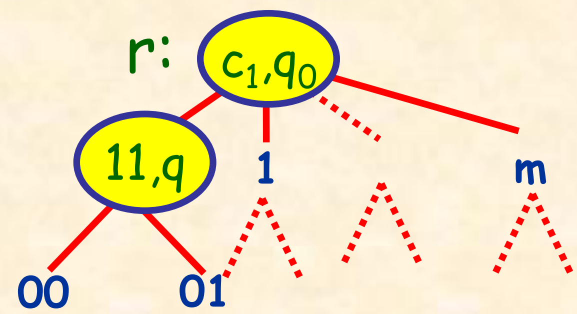


Runs for FEA

- For a FEA A with a transition δ: $Q \times \Sigma \rightarrow B^+(D_b \times Q)$
- A run over a forest $\langle F,V \rangle$ is a $(F \times Q)$–labeled tree, built in accordance with δ and $r(\varepsilon) = (c, q_0)$, for a root c of F.
- Let $r(0)=(11,q)$, $V(11)=a$, and

$$\delta(q,a) = (-1,q_1) \wedge ((\langle root \rangle,q_2) \vee ([root],q_3))$$



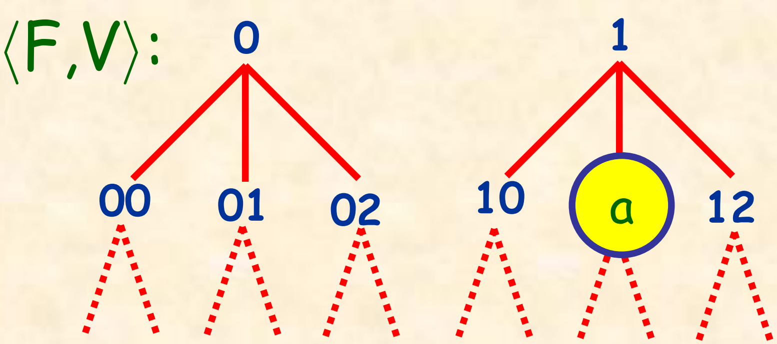


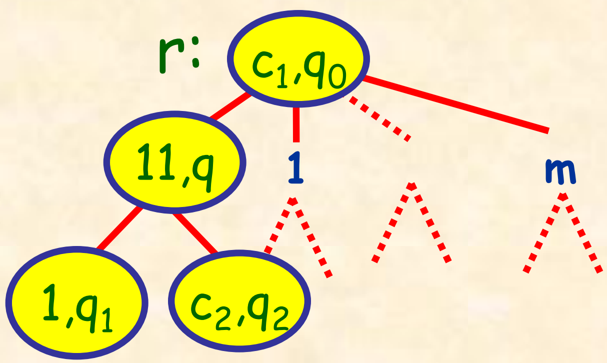
Runs for FEA

- For a FEA A with a transition δ: $Q \times \Sigma \rightarrow B^{+}(D_b \times Q)$
- A run over a forest $\langle F,V \rangle$ is a $(F \times Q)$–labeled tree, built in accordance with δ and $r(\varepsilon) = (c, q_0)$, for a root c of F.
- Let $r(0)=(11,q)$, $V(11)=a$, and

$$\delta(q,a) = (-1,q_1) \wedge ((\langle root \rangle,q_2) \vee ([root],q_3))$$

- Let $S= \{(-1,q_1), (\langle root \rangle,q_2)\}$.



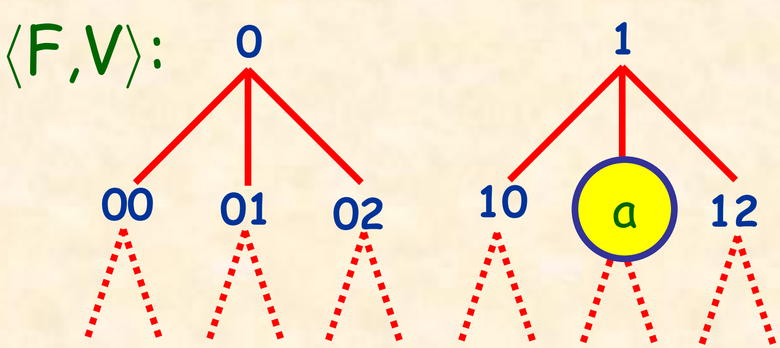


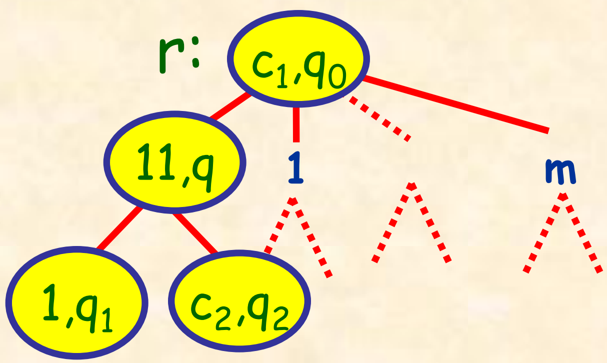
Runs for FEA

- For a FEA A with a transition δ: $Q \times \Sigma \rightarrow B^+(D_b \times Q)$
- A run over a forest $\langle F,V \rangle$ is a $(F\times Q)$–labeled tree, built in accordance with δ and $r(\varepsilon) = (c, q_0)$, for a root c of F.
- Let $r(0)=(11,q)$, $V(11)=a$, and

$$\delta(q,a) = (-1,q_1) \wedge ((\langle root \rangle,q_2)\vee([root],q_3))$$

- Let $S= \{(-1,q_1), (\langle root \rangle,q_2)\}$.





- We use a parity condition.

Acceptance conditions

- Büchi condition: $F \subseteq Q$. A run r is accepting iff for every path, there exists a final state appearing infinitely often
- Formally, a run is accepting if for each path π, **$Inf(r|\pi) \cap F \neq \emptyset$**

Acceptance conditions

- Büchi condition: $F \subseteq Q$. A run r is accepting iff for every path, there exists a final state appearing infinitely often
- Formally, a run is accepting if for each path π, **$Inf(r|\pi) \cap F \neq \emptyset$**

- Parity condition: $\mathbf{F} = \{F_1,\dots,F_m\}$. A run r is accepting if for each path π in r the minimal i for which $Inf(r|\pi) \cap \mathbf{F} \neq \emptyset$ is even

Acceptance conditions

- **Büchi condition:** $F \subseteq Q$. A run r is accepting iff for every path, there exists a final state appearing infinitely often
- Formally, a run is accepting if for each path π, $\mathbf{Inf(r|\pi) \cap F \neq \emptyset}$

- **Parity condition:** $\mathbf{F} = \{F_1,\ldots,F_m\}$. A run r is accepting if for each path π in r the minimal i for which $Inf(r|\pi) \cap \mathbf{F} \neq \emptyset$ is even

- Emptiness:
  - Nondeterministic Buchi Tree Automata (NBT) : PTime-Complete
  - Alternating Buchi Tree Automata (ABT) : ExpTime-Complete
  - Nondeterministic Parity Tree Automata (NPT) : UP ∩ Co-UP
  - Alternating Parity Tree Automata (APT) : ExpTime-Complete

Solving the satisfiability problem

- We show that the satisfiability problem for enriched μ-calculus formulas (except for fully enriched ones) is EXPTime-Complete

Solving the satisfiability problem

- We show that the satisfiability problem for enriched μ-calculus formulas (except for fully enriched ones) is EXPTime-Complete
- Lower Bound: Satisfiability for the μ-calculus is EXPTime-hard [Fisher Ladner 1979]

Solving the satisfiability problem

- We show that the satisfiability problem for enriched μ-calculus formulas (except for fully enriched ones) is EXPTime-Complete
- Lower Bound: Satisfiability for the μ-calculus is EXPTime-hard [Fisher Ladner 1979]
- Upper Bound: We use an automata-theoretic approach:
- Given a sentence φ of the full graded μ-calculus that has m atleast sub-sentences and counts up to b, we can construct a FEA A_φ that
  - accepts the set of tree models of φ with degree at most m(b+1), and
  - has $|\varphi|$ states, index $|\varphi|$.

Solving the satisfiability problem

- We show that the satisfiability problem for enriched μ-calculus formulas (except for fully enriched ones) is EXPTime-Complete
- Lower Bound: Satisfiability for the μ-calculus is EXPTime-hard [Fisher Ladner 1979]
- Upper Bound: We use an automata-theoretic approach:
- Given a sentence φ of the full graded μ-calculus that has m atleast sub-sentences and counts up to b, we can construct a FEA A_φ that
  - accepts the set of tree models of φ with degree at most m(b+1), and
  - has |φ| states, index |φ|.
- Given a sentence φ of the hybrid graded/full μ-calculus with m atleast subsentences, k nominals, and counts up to b, we can built a FEA A_φ that
  - accepts all quasi forest models of φ with degree max{k+1, m(b+1)}, and
  - has $O(|\varphi|^2)$ states, index |φ|.

Solving the satisfiability problem

- We show that the satisfiability problem for enriched μ-calculus formulas (except for fully enriched ones) is EXPTime-Complete
- Lower Bound: Satisfiability for the μ-calculus is EXPTime-hard [Fisher Ladner 1979]
- Upper Bound: We use an automata-theoretic approach:
- Given a sentence φ of the full graded μ-calculus that has m atleast sub-sentences and counts up to b, we can construct a FEA A_φ that
  - accepts the set of tree models of φ with degree at most m(b+1), and
  - has $|\varphi|$ states, index $|\varphi|$.
- Given a sentence φ of the hybrid graded/full μ-calculus with m atleast subsentences, k nominals, and counts up to b, we can built a FEA A_φ that
  - accepts all quasi forest models of φ with degree max{k+1, m(b+1)}, and
  - has $O(|\varphi|^2)$ states, index $|\varphi|$.
- In both cases, φ is satisfiable if $L(A_\varphi) \neq \emptyset$

Solving the emptiness problem

- We first reduce the emptiness problem for FEA to the emptiness problem for 2GAPTs.
  - A 2GAPT is a FEA that accepts trees and cannot jump to the root of the input tree.

Solving the emptiness problem

- We first reduce the emptiness problem for FEA to the emptiness problem for 2GAPTs.
  - A 2GAPT is a FEA that accepts trees and cannot jump to the root of the input tree.
- To decide the emptiness of 2GAPTs, we use a reduction to the emptiness problem of GNPT, via "strategy trees"
  - To remove alternation, we build special trees that allow encoding the original run in one having the same tree structure as the input tree.
  - To restrict to unidirectional paths, we use the notion of annotation that allow to decompose each path into downward paths and detours.

Solving the emptiness problem

- We first reduce the emptiness problem for FEA to the emptiness problem for 2GAPTs.
  - A 2GAPT is a FEA that accepts trees and cannot jump to the root of the input tree.
- To decide the emptiness of 2GAPTs, we use a reduction to the emptiness problem of GNPT, via "strategy trees"
  - To remove alternation, we build special trees that allow encoding the original run in one having the same tree structure as the input tree.
  - To restrict to unidirectional paths, we use the notion of annotation that allow to decompose each path into downward paths and detours.
- The result follows from the blow-up involved in building the GNPT and from the complexity for checking its emptiness.

A strategy tree with detour

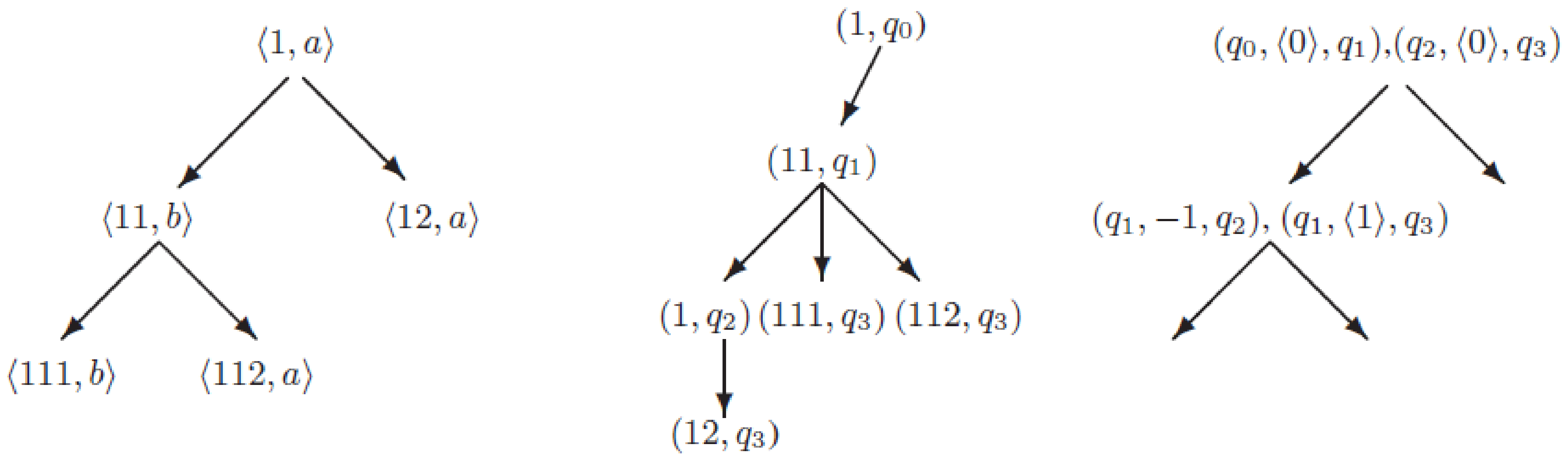


Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.

A strategy tree with detour

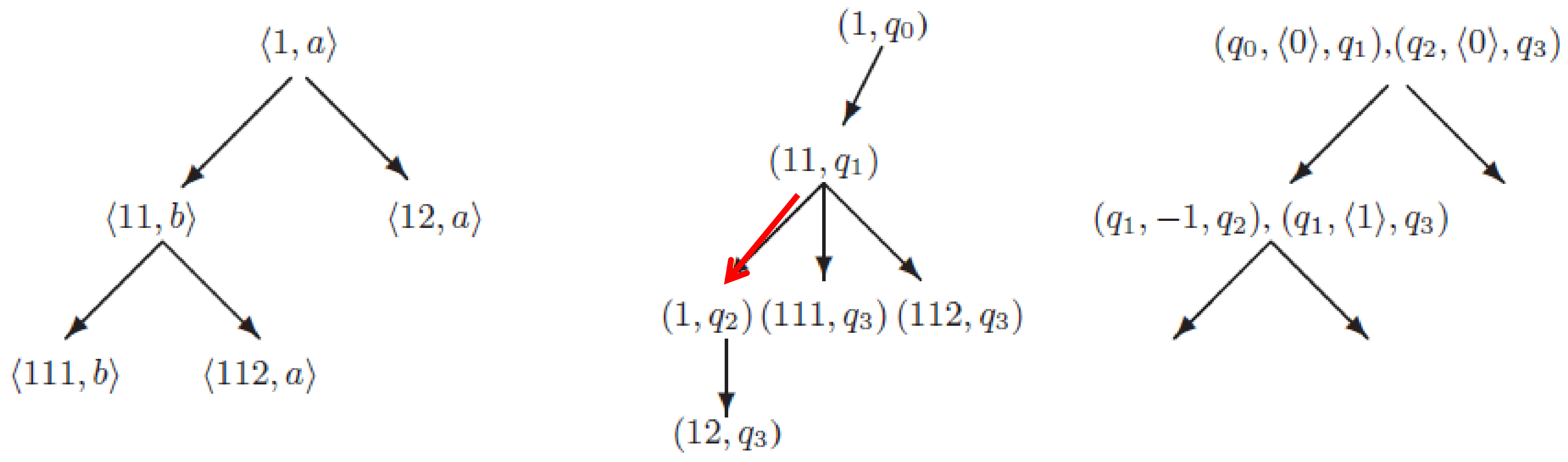


Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.

A strategy tree with detour

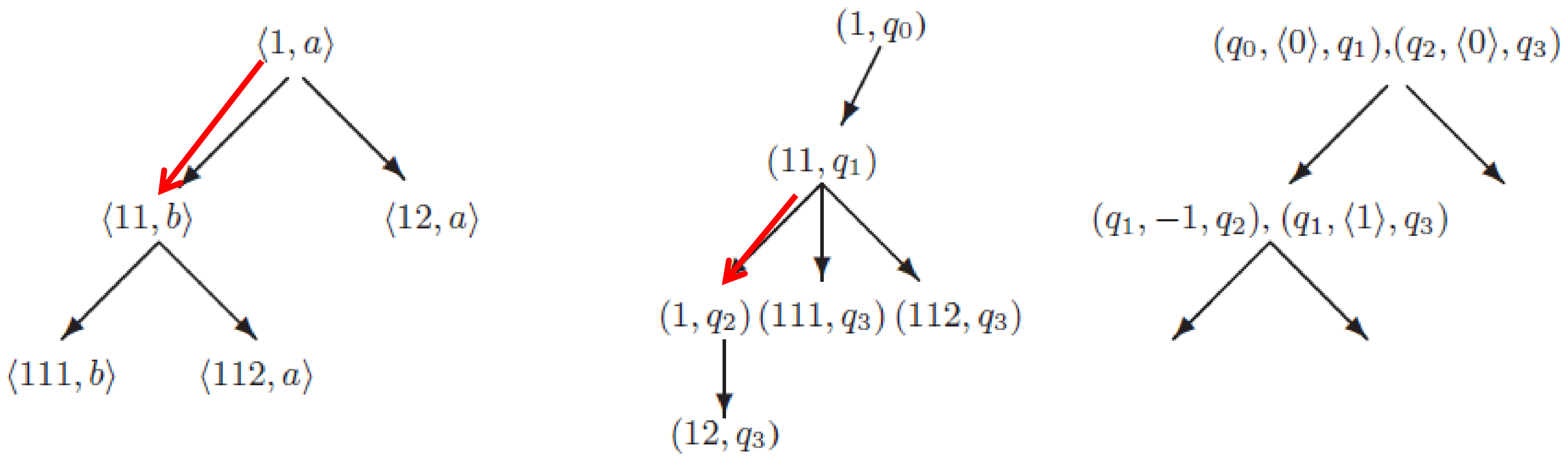


Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.

A strategy tree with detour

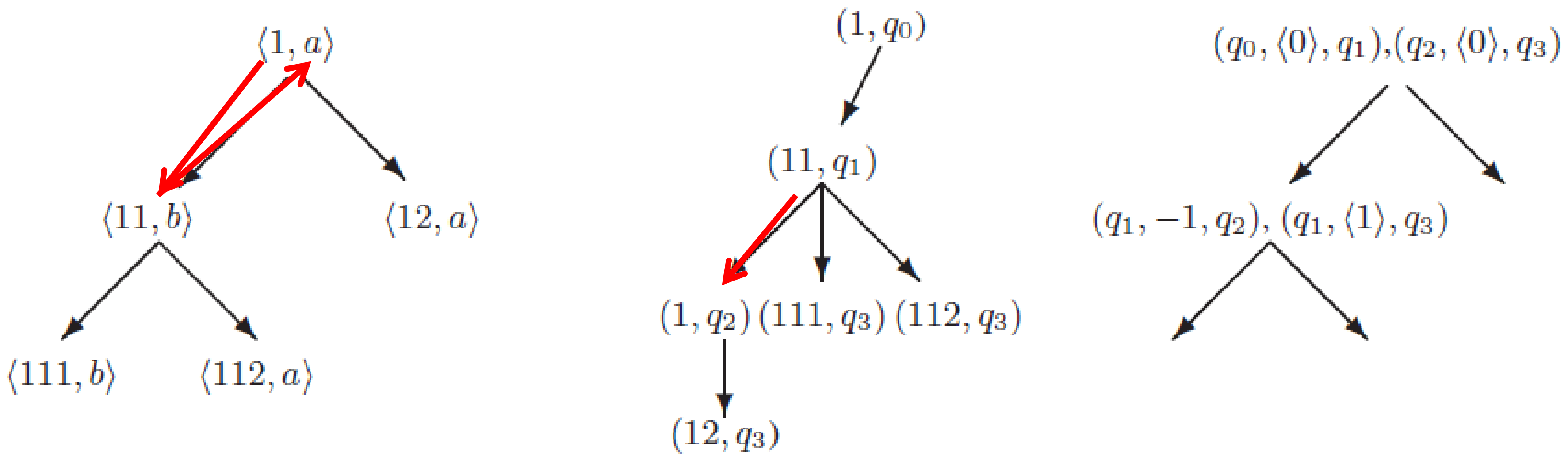


Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.

A strategy tree with detour

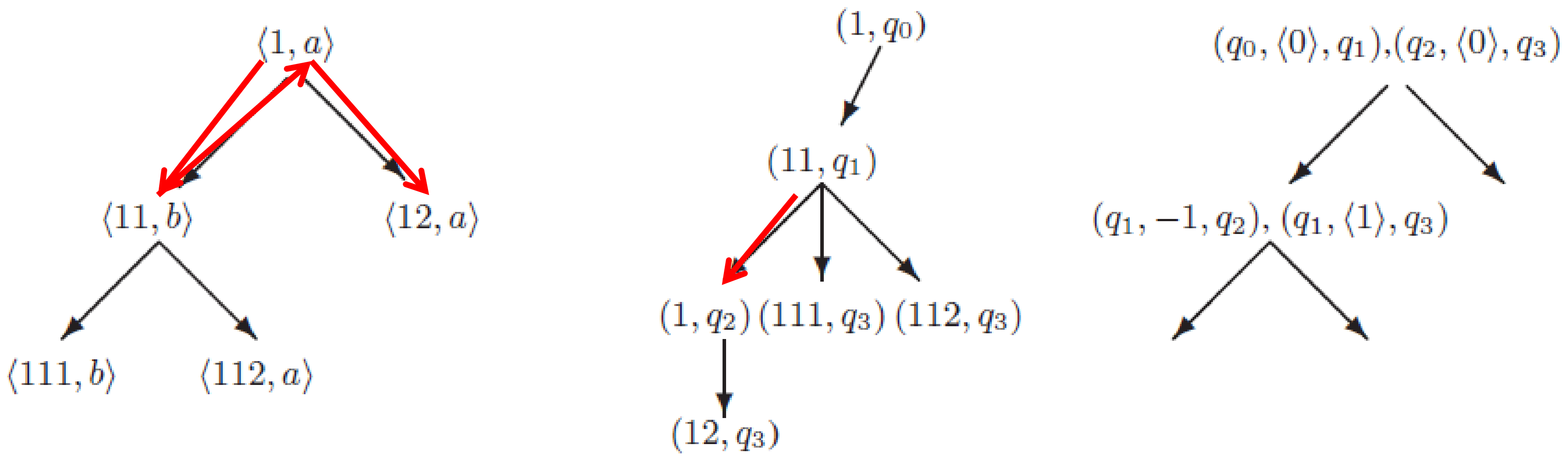


Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.

A strategy tree with detour

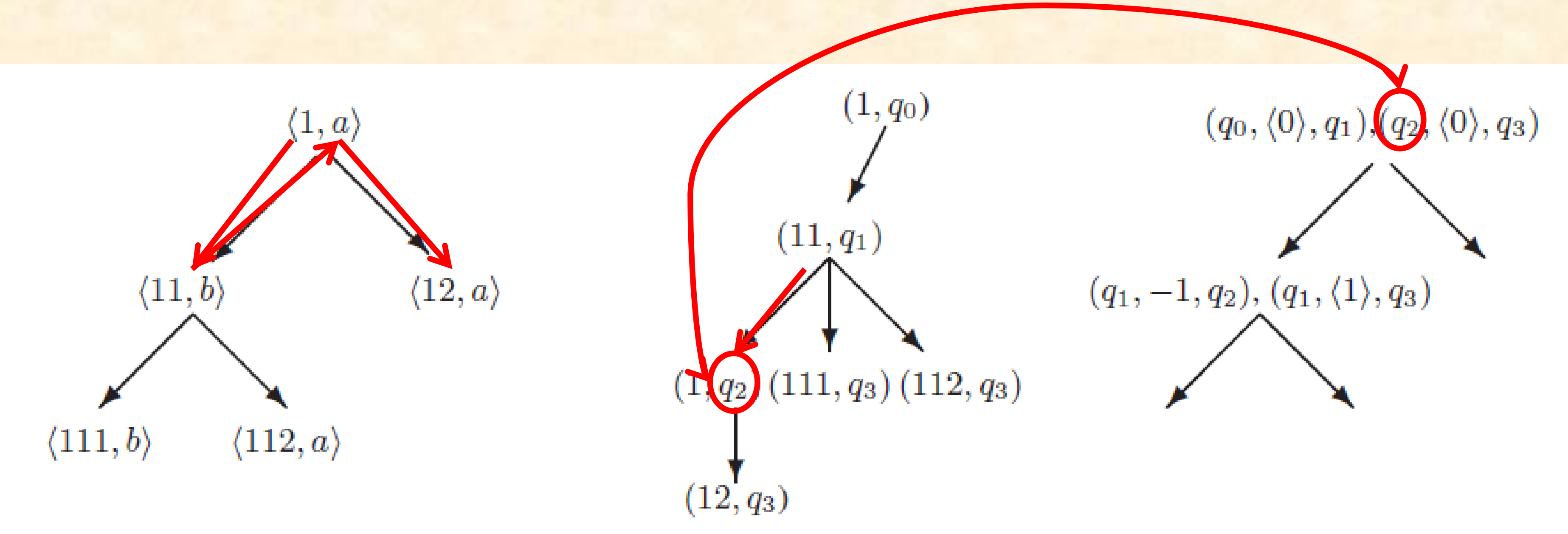


Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.

A Summary for Enriched μ-calculi

Results on the satisfiability problem for Enriched μ-calculi				
	Inverse programs	Graded modalities	Nominals	Complexity
fully enriched	×	×	×	Undecidable[1]
full hybrid	×		×	ExpTime[2]
full graded	×	×		
hybrid graded		×	×	
graded		×		ExpTime 1ary/2ary[3]
full	×			ExpTime[5]

1. [Bonatti, Peron 2004]
2. [Sattler, Vardi 2001]
3. [Vardi 1998]
4. [Calvanese, De Giacomo, Lenzerini, 2001]
5. [Kupferman, Sattler, Vardi, 2002]

A Summary for Enriched μ-calculi

Results on the satisfiability problem for Enriched μ-calculi				
	Inverse programs	Graded modalities	Nominals	Complexity
fully enriched	x	x	x	Undecidable[1]
full hybrid	x		x	ExpTime[2]
full graded	x	x		ExpTime 2ary (1ary[4])
hybrid graded		x	x	
graded		x		ExpTime 1ary/2ary[3]
full	x			ExpTime[5]

1. [Bonatti, Peron 2004]
2. [Sattler, Vardi 2001]
3. [Vardi 1998]
4. [Calvanese, De Giacomo, Lenzerini, 2001]
5. [Kupferman, Sattler, Vardi, 2002]

A Summary for Enriched μ-calculi

Results on the satisfiability problem for Enriched μ-calculi				
	Inverse programs	Graded modalities	Nominals	Complexity
fully enriched	×	×	×	Undecidable[1]
full hybrid	×		×	ExpTime[2]
full graded	×	×		ExpTime 2ary (1ary[4])
hybrid graded		**×**	×	**ExpTime 1ary/2ary**
graded		**×**		ExpTime 1ary/2ary[3]
full	×			ExpTime[5]

1. [Bonatti, Peron 2004]
2. [Sattler, Vardi 2001]
3. [Vardi 1998]
4. [Calvanese, De Giacomo, Lenzerini, 2001]
5. [Kupferman, Sattler, Vardi, 2002]

Enriching Temporal Logics

- μ-calculus is a very expressive but too low-level logic.
- Branching time temporal logics such as CTL, and CTL* are less expressive but much more human-friendly.

Enriching Temporal Logics

- μ-calculus is a very expressive but too low-level logic.
- Branching time temporal logics such as CTL, and CTL* are less expressive but much more human-friendly.
- What about enriching CTL and CTL* with graded modalities.
  - So far, only CTL has been fully solved, both in unary and binary coding.
  - Graded CTL is exponentially more succinct than graded μ-calculus.
  - The satisfiability problem remains ExpTime-Complete

Enriching Temporal Logics

- μ-calculus is a very expressive but too low-level logic.
- Branching time temporal logics such as CTL, and CTL* are less expressive but much more human-friendly.
- What about enriching CTL and CTL* with graded modalities.
  - So far, only CTL has been fully solved, both in unary and binary coding.
  - Graded CTL is exponentially more succinct than graded μ-calculus.
  - The satisfiability problem remains ExpTime-Complete
- Moving from μ-calculus to CTL with graded modalities, we need to move from graded world modalities to graded path modalities!

Syntax of GCTL* and GCTL

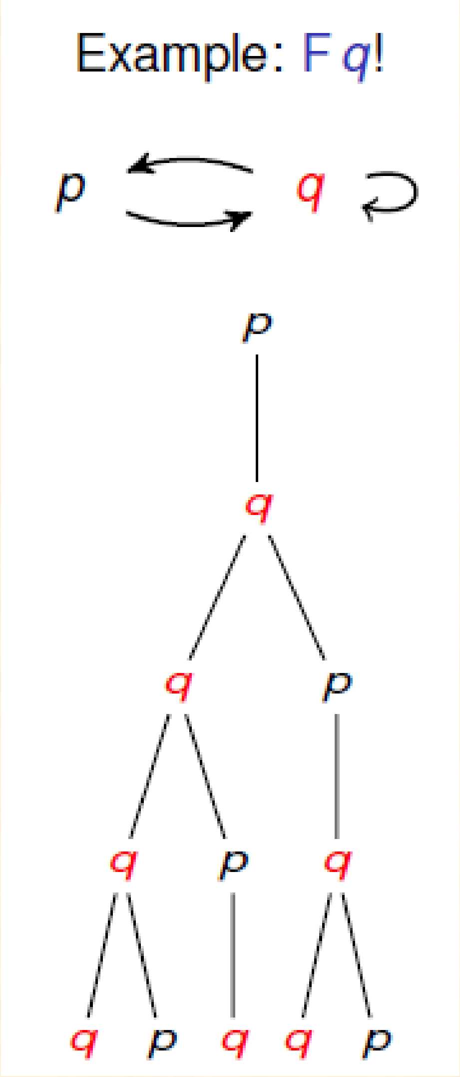
- GCTL* extends CTL* with new graded path quantifiers:
  - "there exists at least n paths satisfying a given property";
  - "all but at most n paths satisfy a given property".

Syntax of GCTL* and GCTL

- GCTL* extends CTL* with new graded path quantifiers:
  - "there exists at least n paths satisfying a given property";
  - "all but at most n paths satisfy a given property".
- CTL* uses state and path formulas built inductively as follows:
- State-formulas:
  - $\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid E^{\geq n}\, \psi \mid A^{<n}\, \psi$
  - where $p \in AP$ and ψ is a path-formula
- path-formulas (LTL):
  - $\psi := \varphi \mid \psi \wedge \psi \mid \neg\psi \mid X\psi \mid \psi\, U\, \psi$
  - where φ is a state-formula, and ψ a path-formula
- GCTL formulas are obtained by forcing each temporal operator to be coupled with a path quantifier

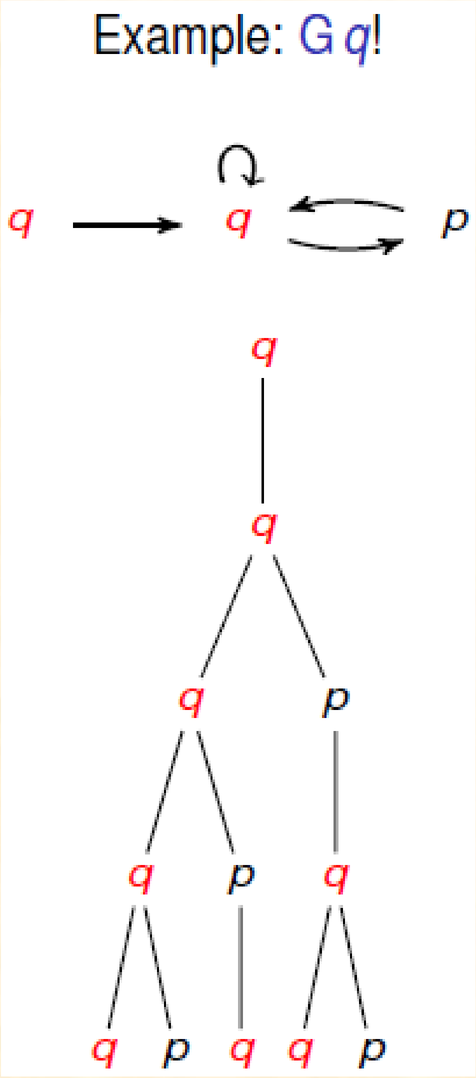
Counting paths

- What does counting paths mean?
  - A property ensured by a common prefix may be satisfied on an infinite number of paths.



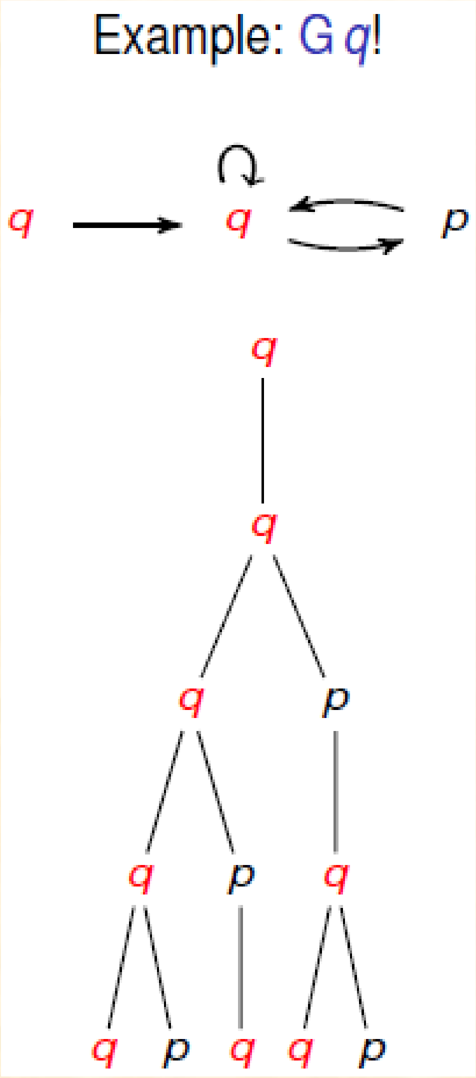
Counting paths

- What does counting paths mean?
  - *A property ensured by a common prefix may be satisfied on an infinite number of paths.*
  - It may happen that the prefix satisfies a formula but a whole path may not.



Counting paths

- What does counting paths mean?
  - A property ensured by a common prefix may be satisfied on an infinite number of paths.
  - It may happen that the prefix satisfies a formula but a whole path may not.
- We restrict to minimal and conservative paths
- Two paths are equivalent if
  - their common prefix satisfy the formula.
  - no matter how these prefixes are extended in the structure, the paths satisfy the formula.



Semantics of GCTL*

- For a Kripke structure K, a world w, and a GCTL* path formula ψ,
- Let P(K, w, ψ) be the set of minimal and conservative paths of K starting in w and satisfying ψ

Semantics of GCTL*

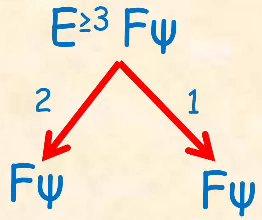
- For a Kripke structure K, a world w, and a GCTL* path formula ψ,
- Let P(K, w, ψ) be the set of minimal and conservative paths of K starting in w and satisfying ψ
  - $K, w \models E^{\geq n} \psi$ iff $|P(K, w, \psi)| \geq n$
  - $K, w \models A^{<n} \psi$ iff $|P(K, w, \neg\psi)| < n$
- For n=1, we write Eψ and Aψ instead of $E^{\geq 1} \psi$ e $A^{<1} \psi$

Solving GCTL in unary coding

- Let ψ be a GCTL formula with grades coded in unary.
- From ψ we build in linear time a "Partitioning Alternating Büchi Tree Automata" (PABT) P_{ψ}

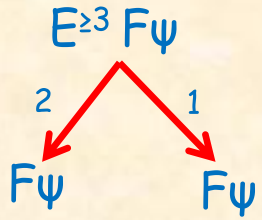
Solving GCTL in unary coding

- Let ψ be a GCTL formula with grades coded in unary.
- From ψ we build in linear time a "Partitioning Alternating Büchi Tree Automata" (PABT) P_ψ
- A PABT accepts all tre models of a formula, by «gessing» how to partition a required graded modality among successors



Solving GCTL in unary coding

- Let ψ be a GCTL formula with grades coded in unary.
- From ψ we build in linear time a "Partitioning Alternating Büchi Tree Automata" (PABT) P_ψ
- A PABT accepts all tre models of a formula, by «gessing» how to partition a required graded modality among successors



- By means of an opportune extension of the Myhano-Hayashi tecnique, we translate in Exponential Time P_ψ in an NBT B_ψ
- Since the emptiness of $L(B_\psi)$ can be checked in polynomial time, we get that the satisfiability problem for GCTL is in ExpTime.
- ExpTime hardness comes from the satisfiability problem for CTL

Solving GCTL in binary coding

- If we use the unary case approach, we lose an exponent:
  - The tree model property requires trees with a branching degree exponential in the highest graded $\mathbf{b}_{max}$ of the formula.

Solving GCTL in binary coding

- If we use the unary case approach, we lose an exponent:
  - The tree model property requires trees with a branching degree exponential in the highest graded $\mathbf{b}_{max}$ of the formula.
- We use a binary encoding of each tree model and split the automata construction into a linear PABT plus a satellite NBT automaton.
  - The tree encoding turns each level of the tree in a binary tree, i.e., brothers of a node become its successors.
  - The satellite is an (exponential) NBT and ensures that each tree model satisfies some structural properties along its paths.

Solving GCTL in binary coding

- If we use the unary case approach, we lose an exponent:
  - The tree model property requires trees with a branching degree exponential in the highest graded $\mathbf{b}_{max}$ of the formula.
- We use a binary encoding of each tree model and split the automata construction into a linear PABT plus a satellite NBT automaton.
  - The tree encoding turns each level of the tree in a binary tree, i.e., brothers of a node become its successors.
  - The satellite is an (exponential) NBT and ensures that each tree model satisfies some structural properties along its paths.
- As the satellite automaton is already an NBT, this avoids to inject an extra exponent when moving both automata to a unique NBT.
- Thus, also in the binary coding, the satisfiability question for GCTL is ExpTime-complete

What about GCTL*

- Solving graded CTL* is even more appealing.
- There are several question to investigate.
- Is GCTL* more succinct than Graded mu-calculus?
- What about the satisfiability?
  - Using a slight variation of the previous reasoning used for GCTL, we get a 3ExpTime upper bound.
  - As CTL* satisfiability is 2ExpTime-complete, it is an open question to decide the exact complexity of the problem for GCTL*

Further directions about GCTL and GCTL*

- What about GCTL/ GCTL* plus backwords modalities?
- CTL and CTL* have been investigated with respect to (linear and branching) Past modalities.
- PCTL (PCTL*) is (2)ExpTime-complete.
- What about GCTL/GCTL over more enriched structures: Hierachical, pushown, weighted etc…

Enriched modalities vs. open systems

- Enriched mu-calculi has been investigated in the setting of module checking.
- Same results as in the satisfiability case:
  - Undecidable if we consider the fully enriched mu-calculus.
  - ExpTime-complete for every fragment.

Conclusion

Results on the satisfiability problem for Enriched μ-calculi				
	Inverse programs	Graded modalities	Nominals	Complexity
fully enriched	x	x	x	Undecidable[1]
full hybrid	x		x	ExpTime[2]
full graded	x	x		ExpTime 2ary (1ary[4])
hybrid graded		**x**	x	**ExpTime 1ary/2ary**
graded		**x**		ExpTime 1ary/2ary[3]
full	x			ExpTime[5]
Results on the satisfiability problem for GCTL				
GCTL		**x**		**ExpTime 1ary/2ary**
Past CTL	x			ExpTime[6]

1. [Bonatti, Peron 2004]
2. [Sattler, Vardi 2001]
3. [Vardi 1998]
4. [Calvanese, De Giacomo, Lenzerini, 2001]
5. [Kupferman, Sattler, Vardi, 2002]
6. [Kupferman, Pnueli 1995]

References

- P.A. Bonatti, C. Lutz, A. Murano, M. Y.Vardi. The complexity of Enriched mu-calculi. Lecture methods in Computer Science 2008
  - Invited extended version of ICALP'06
- A. Bianco, F. Mogavero, A. Murano. Graded Computation Tree Logic. ACM Transaction of Computation Logic 2012
  - Extended version of LICS'09 and CSL'10
- A. Ferrante, A. Murano, M. Parente. Enriched mu-calculi module checking. Lecture methods in Computer Science 2008
  - Invited extended version of FOSSACS '07 and LPAR'07

Thank you for your attention!