Advanced Automata Theory

SS 2017

Helmut Seidl

Overview over this Lecture

Part 1: Tree automata for Program Analysis

- functional languages
- logic languages
- cryptographic protocols

Overview over this Lecture

Part 1: Tree automata for Program Analysis

- functional languages
- logic languages
- cryptographic protocols

Part 2: Type Checking for XML Transformations

- Types for XML
- XML Transformations
- Decomposition of XML Transformations

Overview over this Lecture

Part 1: Tree automata for Program Analysis

- functional languages
- logic languages
- cryptographic protocols

Part 2: Type Checking for XML Transformations

- Types for XML
- XML Transformations
- Decomposition of XML Transformations

Part 3: Equivalence Problems

- Straight-line Programs
- Topdown Tree-to-tree Transformations
- Topdown Tree-to-string Transformations

Automaton

- accepts structures
- defines a predicate on structures, or equivalently,
- defines a set of structures

Automaton

- accepts structures
- defines a predicate on structures, or equivalently,
- defines a set of structures

Automata, here: finite-state

⫽ easy to understand
⫽ decidability/tractability
⫽ normal forms
⫽ learning
⫽ equivalence

Examples of structures

Examples of structures

words

- finite labeled
  - compiler construction (scanners)
  - string processing, searching
- infinite labeled
  - system behaviors - linear-time logic

Examples of structures

words

- finite labeled
  - compiler construction (scanners)
  - string processing, searching
- infinite labeled
  - system behaviors - linear-time logic

trees

- finite ranked ordered labeled
  - syntax trees
  - terms
- finite unranked ordered labeled
  - XML, JSON
- infinite ranked unordered labeled
  - system behaviors - branching-time logic
- infinite ranked ordered labeled
  - monadic second order logic

Transducer

realizes a function/relation on structures.

Variations

- string-to-string (classical)
- tree-to-tree
  - ⫽ program transformations
  - ⫽ NL translations
  - ⫽ syntax-directed computation
- tree-to-string
  - ⫽ XML/JSON transformations

Part 1

Tree Automata for Program Analysis

Motivation

- Program analysis tries to statically infer properties of the runtime behavior of a program,

Motivation

- Program analysis tries to statically infer properties of the runtime behavior of a program, e.g.,
  - values of variables;
  - reachable configurations.

Motivation

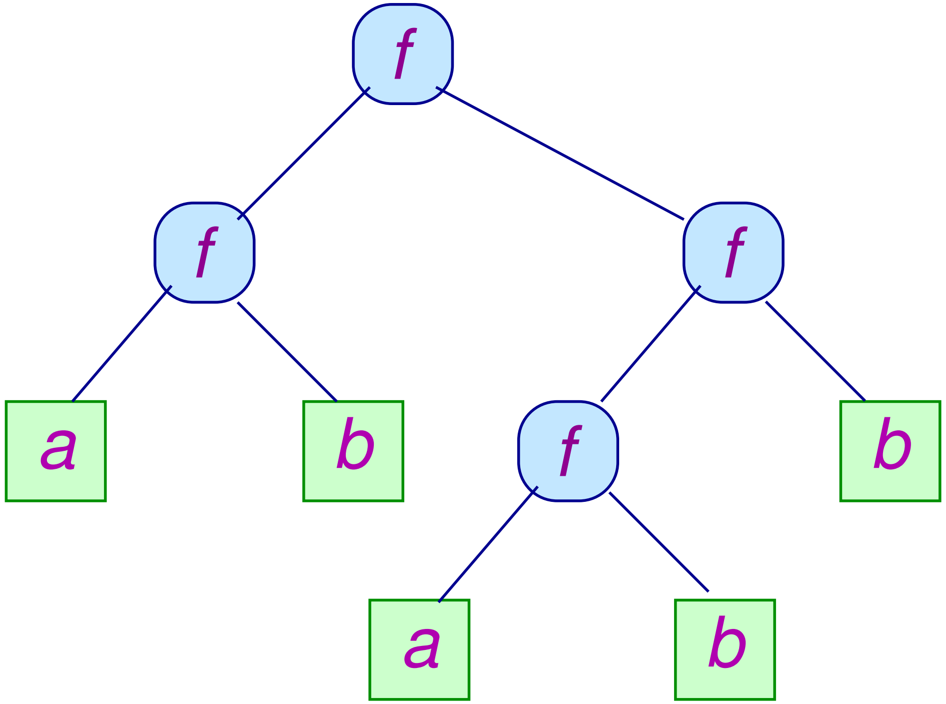
- Program analysis tries to statically infer properties of the runtime behavior of a program, e.g.,
  - values of variables;
  - reachable configurations.
- Often, such analyses result in tree automata.

Motivation

- Program analysis tries to statically infer properties of the runtime behavior of a program, e.g.,
  - values of variables;
  - reachable configurations.
- Often, such analyses result in tree automata.
- A formalism is required to conveniently express and perform operations on tree automata.

0. Basics

A Tree



Trees

Properties

- ranked ordered
- labeled
- finite

Trees

Properties

- ranked ordered
- labeled
- finite

$$= \quad \text{terms}$$

Automata

- Tree automaton A generalizes word automaton.
- A run of A on tree t is a mapping of the nodes of t to states

Automata

- Tree automaton A generalizes word automaton.
- A run of A on tree t is a mapping of the nodes of t to states ...

  which locally respects the transition relation

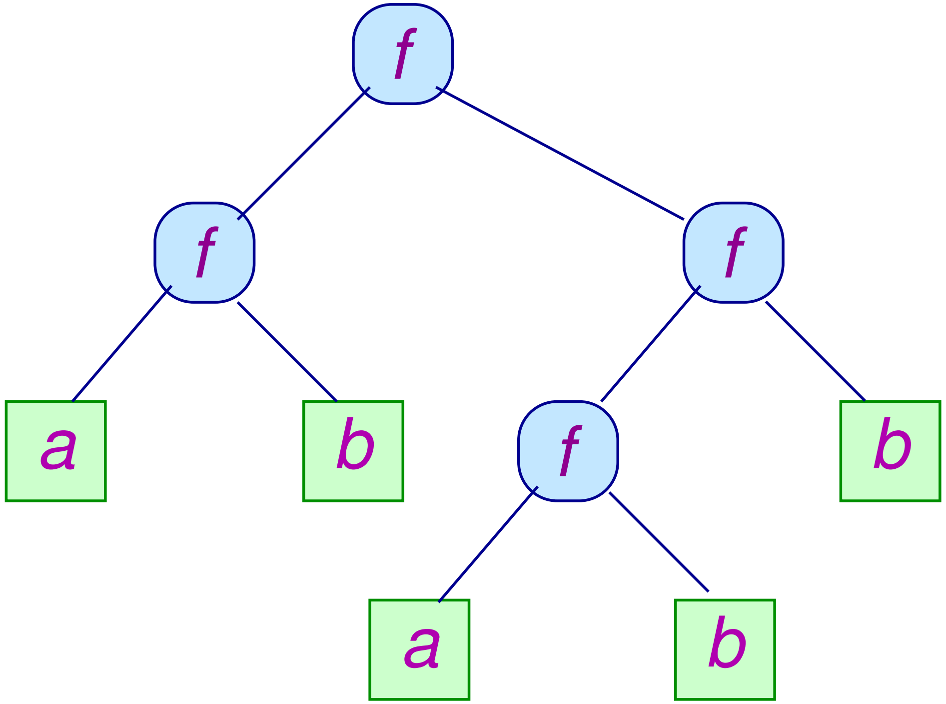
Automata

- Tree automaton A generalizes word automaton.
- A run of A on tree t is a mapping of the nodes of t to states ...

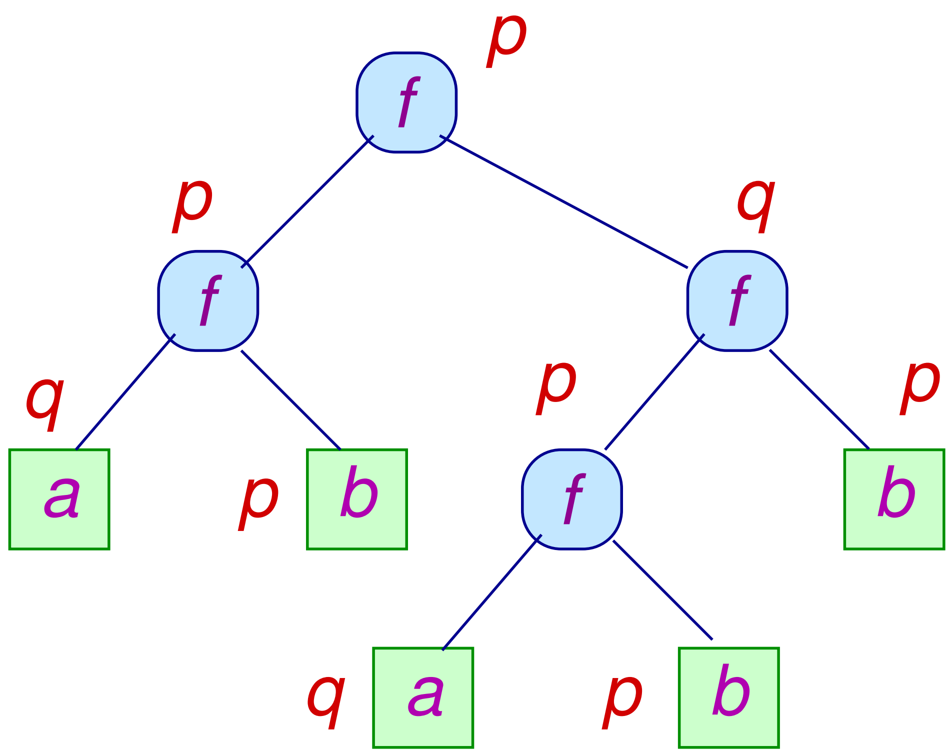
  which locally respects the transition relation

$$\delta \subseteq \bigcup_{j \geq 0} Q \times \Sigma_j \times Q^j$$

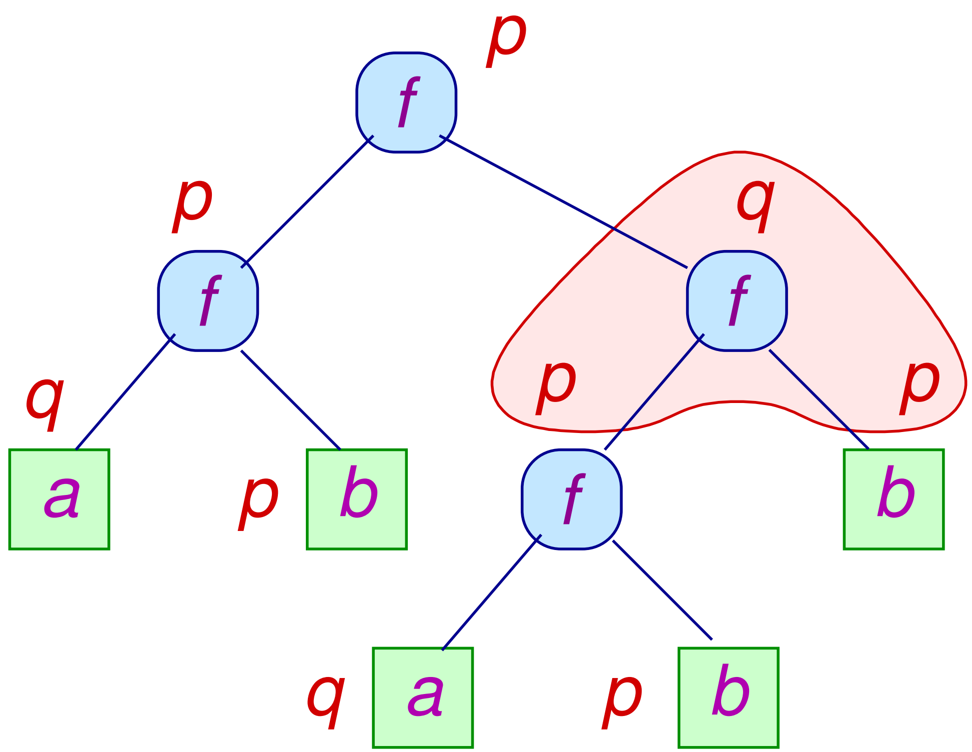
A Run



A Run



A Run



An Example Automaton

$A = (Q, \Sigma, \delta, F)$ where

Q	$=$	$\{p, q\}$	//	set of states
$F \subseteq Q$	$=$	$\{p\}$	//	accepting states
Σ_0	$=$	$\{a, b\}$	//	input alphabet of ranl
Σ_2	$=$	$\{f\}$	//	input alphabet of ranl
δ	$=$	$\{\ (q, a), (p, b),$ $(p, f, qp), (p, f, pq),$ $(q, f, qq), (q, f, pp)\ \}$	//	transitions

Accepting Run

A run is accepting if it assigns an accepting state to the root.

Accepting Run

A run is accepting if it assigns an accepting state to the root.

The language $\mathcal{L}(A)$ of a tree automaton A is the set of trees for which there is an accepting run of A.

Accepting Run

A run is accepting if it assigns an accepting state to the root.

The language $\mathcal{L}(A)$ of a tree automaton A is the set of trees for which there is an accepting run of A.

A language T is regular if $T = \mathcal{L}(A)$ for some tree automaton A.

Clauses

Alternative representation:

state	unary predicate
symbol	constructor
transition	Horn clause

Clauses

Alternative representation:

state	unary predicate
symbol	constructor
transition	Horn clause

$$
\begin{array}{lcl}
q(a) & \Leftarrow & \\
p(b) & \Leftarrow & \\
p(f(X, Y)) & \Leftarrow & q(X), p(Y) \\
p(f(X, Y)) & \Leftarrow & p(X), q(Y) \\
q(f(X, Y)) & \Leftarrow & q(X), q(Y) \\
q(f(X, Y)) & \Leftarrow & p(X), p(Y)
\end{array}
$$

Decision Problems

Decision Problems

Emptiness: linear time, P-complete

Folklore

Decision Problems

Emptiness: linear time, P-complete

Folklore

Tree Problem, fixed automaton: uniform-NC_1-complete
under $DLOG$-reductions

Decision Problems

Emptiness: linear time, P-complete

Folklore

Tree Problem, fixed automaton: uniform-NC_1-complete under *DLOG*-reductions

Tree Problem, uniform: *LOGCFL*-complete under *LOGSPACE*-reductions

Lohrey, RTA2001

Decision Problems

Emptiness: linear time, P-complete

Folklore

Tree Problem, fixed automaton: uniform-NC_1-complete under *DLOG*-reductions

Tree Problem, uniform: *LOGCFL*-complete under *LOGSPACE*-reductions

Lohrey, RTA2001

Equivalence: *DEXPTIME*-complete under *LOGSPACE*-reductions

S., 1990

Deterministic Automata

- The example TA is (complete and) bottom-up deterministic.

Deterministic Automata

- The example TA is (complete and) bottom-up deterministic.
- For every TA, an equivalent TA can be constructed which is bottom-up deterministic.

Deterministic Automata

- The example TA is (complete and) bottom-up deterministic.
- For every TA, an equivalent TA can be constructed which is bottom-up deterministic.
- The example TA is not top-down deterministic.

Deterministic Automata

- The example TA is (complete and) bottom-up deterministic.
- For every TA, an equivalent TA can be constructed which is bottom-up deterministic.
- The example TA is not top-down deterministic.
- Is there a top-down deterministic TA which is equivalent to the example TA ?

The Powerset Construction

Let $A = (Q, \Sigma, \delta, F)$ denote a TA.

The Powerset Construction

Let $A = (Q, \Sigma, \delta, F)$ denote a TA.

Idea

For each tree t, collect the set $B \subseteq Q$ of states at the root for which there is a run of A.

The Powerset Construction

Let $A = (Q, \Sigma, \delta, F)$ denote a TA.

Idea

For each tree t, collect the set $B \subseteq Q$ of states at the root for which there is a run of A.

Define $\mathcal{P}(A) = (\mathcal{P}(Q), \Sigma, \mathcal{P}(\delta), \mathcal{P}(F))$ where

- $\mathcal{P}(Q)$ is the powerset of Q;
- $\mathcal{P}(F) = \{B \in \mathcal{P}(Q) \mid B \cap F \neq \emptyset\}$
- $(B, f, B_1 \dots B_k) \in \mathcal{P}(\delta)$ iff

$$B = \{q \in Q \mid \exists\, q_1 \in B_1, \dots, , q_k \in B_k.\ (q, f, q_1 \dots q_k) \in \delta\}$$

Then $\mathcal{P}(A)$ is bottom-up deterministic.

Correctness

For every tree t and every subset $B \subseteq Q$, the following statements are equivalent:

1. There is a run of $\mathcal{P}(A)$ for t with B at the root;
2. B equals the set of all $q \in Q$ so that there is a run of A for t with q at the root.

Correctness

For every tree t and every subset $B \subseteq Q$, the following statements are equivalent:

1. There is a run of $\mathcal{P}(A)$ for t with B at the root;
2. B equals the set of all $q \in Q$ so that there is a run of A for t with q at the root.

Proof Induction over the structure of t.

Correctness

For every tree t and every subset $B \subseteq Q$, the following statements are equivalent:

1. There is a run of $\mathcal{P}(A)$ for t with B at the root;
2. B equals the set of all $q \in Q$ so that there is a run of A for t with q at the root.

Proof Induction over the structure of t.

Corollary
$\mathcal{L}(A) = \mathcal{L}(\mathcal{P}(A))$.

Remark

- The construction is inherently exponential.
- A practical implementation will only consider those subsets $B \subseteq Q$ which occur at the root of some tree.

Remark

- The construction is inherently exponential.
- A practical implementation will only consider those subsets $B \subseteq Q$ which occur at the root of some tree.
- What about the topdown constructions ?

The Dual Powerset Construction

Define $\mathcal{P}^\top(A) = (\mathcal{P}^\top(Q), \Sigma, \mathcal{P}^\top(\delta), F)$ where

- $\mathcal{P}^\top(Q)$ is the powerset of Q;
- $(B, a) \in \mathcal{P}^\top(\delta)$ iff $\exists\, q \in B.\ (q, a) \in \delta$;

The Dual Powerset Construction

Define $\mathcal{P}^\top(A) = (\mathcal{P}^\top(Q), \Sigma, \mathcal{P}^\top(\delta), F)$ where

- $\mathcal{P}^\top(Q)$ is the powerset of Q;
- $(B, a) \in \mathcal{P}^\top(\delta)$ iff $\exists\, q \in B.\ (q, a) \in \delta$;
- $(B, f, B_1 \dots B_k) \in \mathcal{P}^\top(\delta)$ iff for $i = 1, \dots, k$,

$$B_i = \{q_i \in Q \mid \exists\, q \in B, q_1, \dots, q_{i-1}, q_{i+1}, \dots, q_k \in Q.\ (q, f, q_1 \dots q_k) \in \delta\}$$

The Dual Powerset Construction

Define $\mathcal{P}^\top(A) = (\mathcal{P}^\top(Q), \Sigma, \mathcal{P}^\top(\delta), F)$ where

- $\mathcal{P}^\top(Q)$ is the powerset of Q;
- $(B, a) \in \mathcal{P}^\top(\delta)$ iff $\exists\, q \in B.\ (q, a) \in \delta$;
- $(B, f, B_1 \dots B_k) \in \mathcal{P}^\top(\delta)$ iff for $i = 1, \dots, k$,

$$B_i = \{q_i \in Q \mid \exists\, q \in B, q_1, \dots, q_{i-1}, q_{i+1}, \dots, q_k \in Q.\ (q, f, q_1 \dots q_k) \in \delta\}$$

This automaton is topdown deterministic (possibly partial).

The Dual Powerset Construction

Define $\mathcal{P}^\top(A) = (\mathcal{P}^\top(Q), \Sigma, \mathcal{P}^\top(\delta), F)$ where

- $\mathcal{P}^\top(Q)$ is the powerset of Q;
- $(B, a) \in \mathcal{P}^\top(\delta)$ iff $\exists\, q \in B.\ (q, a) \in \delta$;
- $(B, f, B_1 \dots B_k) \in \mathcal{P}^\top(\delta)$ iff for $i = 1, \dots, k$,

$$B_i = \{q_i \in Q \mid \exists\, q \in B, q_1, \dots, q_{i-1}, q_{i+1}, \dots, q_k \in Q. \\ (q, f, q_1 \dots q_k) \in \delta\}$$

This automaton is topdown deterministic (possibly partial).

It is not necessarily equivalent to A ...

The Example

$$
\begin{array}{rcl}
Q & = & \{p, q\} \\
\Sigma & = & \{a, b, f\} \\
\delta & = & \{ \ (q, a), (p, b), \\
& & \quad (p, f, qp), (p, f, pq), \\
& & \quad (q, f, qq), (q, f, pp) \ \} \\
F & = & \{p\}
\end{array}
$$

The Example

$$
\begin{array}{rcl}
Q & = & \{p, q\} \\
\Sigma & = & \{a, b, f\} \\
\delta & = & \{\ (q, a), (p, b), \\
& & \quad (p, f, qp), (p, f, pq), \\
& & \quad (q, f, qq), (q, f, pp)\ \} \\
F & = & \{p\}
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{P}^\top(Q) & = & \{\{p\}, \{p, q\}\} \\
\mathcal{P}^\top(\delta) & = & \{\ (\{p, q\}, a), (\{p\}, b), (\{p, q\}, b) \\
& & \quad (\{p\}, f, \{p, q\}\{p, q\}) \\
& & \quad (\{p, q\}, f, \{p, q\}\{p, q\})\ \} \\
q_0 & = & \{p\}
\end{array}
$$

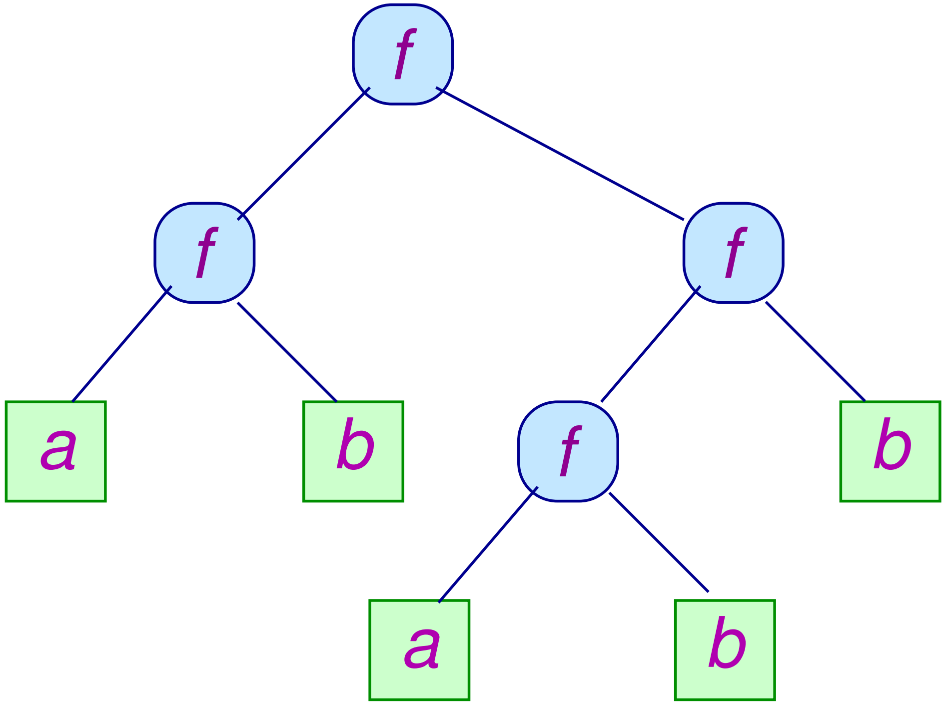
The Example

$$\begin{array}{rcl} Q & = & \{p, q\} \\ \Sigma & = & \{a, b, f\} \\ \delta & = & \{ \ (q, a), (p, b), \\ & & \quad (p, f, qp), (p, f, pq), \\ & & \quad (q, f, qq), (q, f, pp) \ \} \\ F & = & \{p\} \end{array}$$

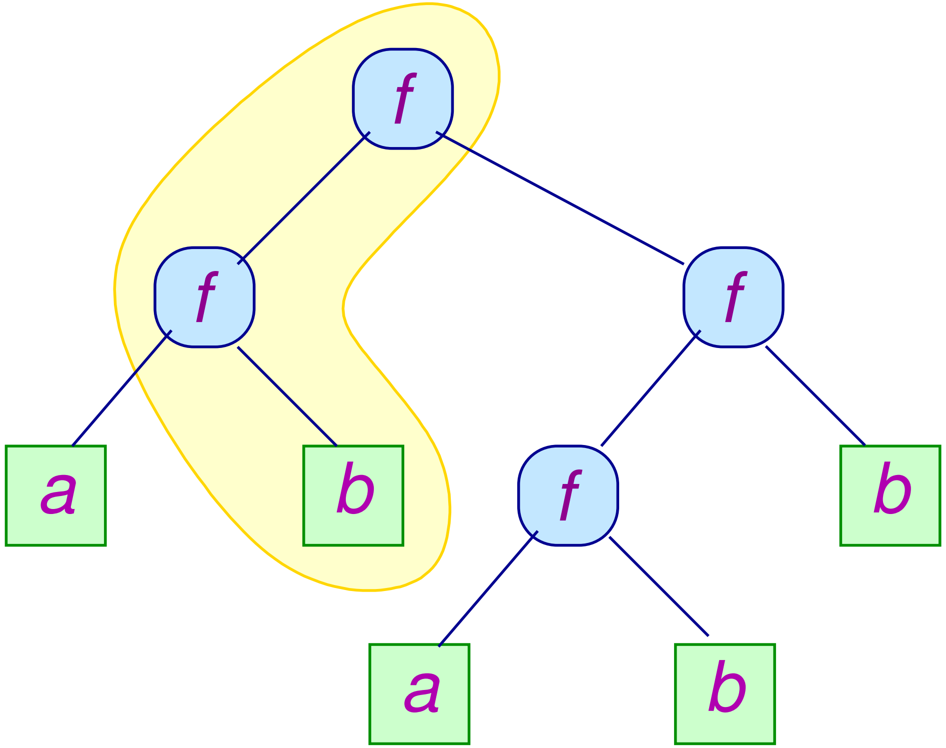
$$\begin{array}{rcl} \mathcal{P}^{\top}(Q) & = & \{\{p\}, \{p, q\}\} \\ \mathcal{P}^{\top}(\delta) & = & \{ \ (\{p, q\}, a), (\{p\}, b), (\{p, q\}, b) \\ & & \quad (\{p\}, f, \{p, q\}\{p, q\}) \\ & & \quad (\{p, q\}, f, \{p, q\}\{p, q\}) \ \} \\ q_0 & = & \{p\} \end{array}$$

... accepts $t = f(a, a)$??

Path



Path



$\langle f, 1\rangle\langle f, 2\rangle\, b$

Homogenuity

L is homogeneous iff

$$t \in L \quad \text{iff} \quad \mathrm{path}(t) \subseteq \mathit{path}(L)$$

Homogenuity

L is homogeneous iff

$$t \in L \quad \text{iff} \quad \text{path}(t) \subseteq \mathit{path}(L)$$

- {f(a,b),f(b,a),f(a,a),f(b,b)} is homogeneous,

Homogenuity

L is homogeneous iff

$$t \in L \quad \text{iff} \quad \text{path}(t) \subseteq \mathit{path}(L)$$

- {f(a,b),f(b,a),f(a,a),f(b,b)} is homogeneous,
- {f(a,b),f(b,a)} is not.