Algorithms for Model Checking (2IW55)

Lecture 10

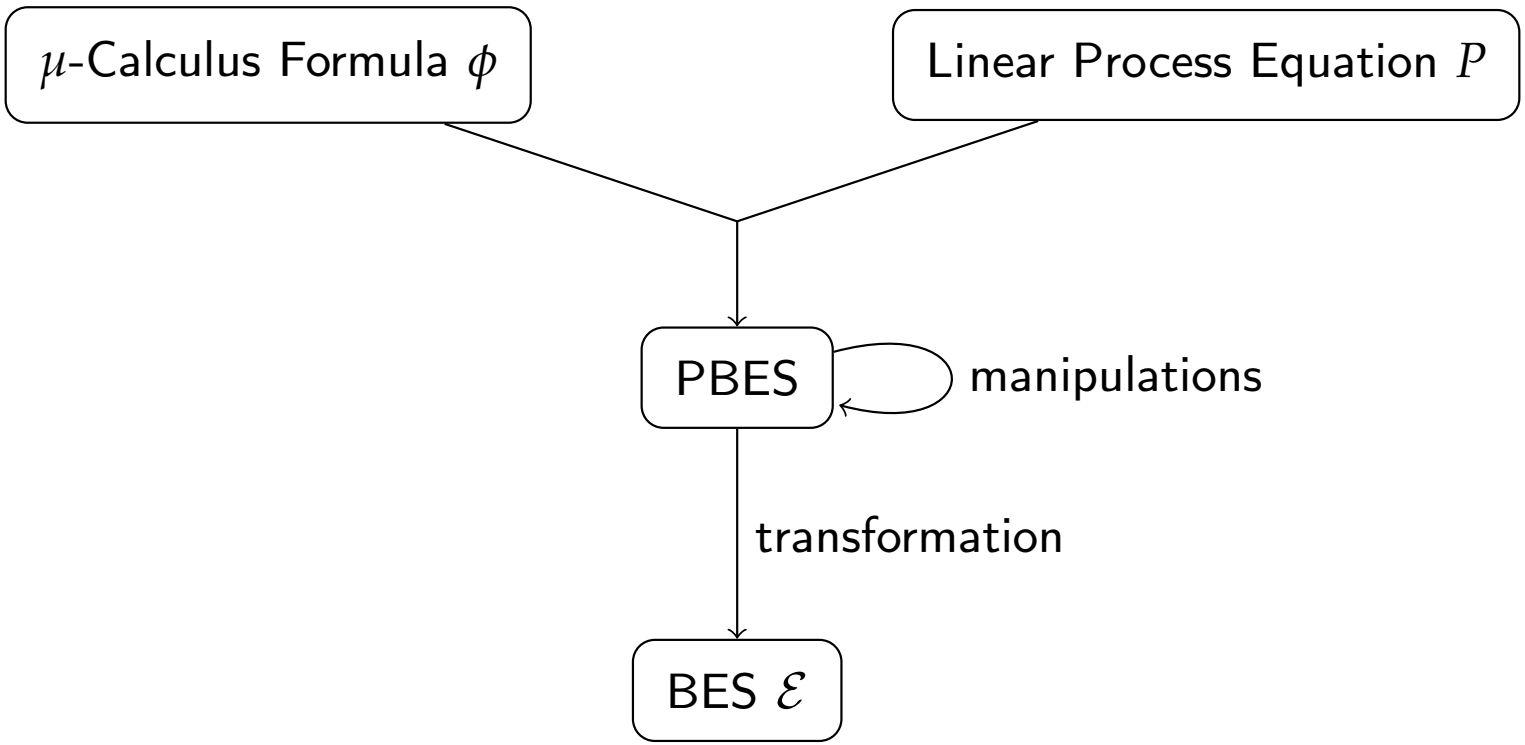
Parameterised Boolean Equation Systems

Tim Willemse
(timw@win.tue.nl)
http://www.win.tue.nl/∼timw
HG 6.81

Outline

Transforming Satisfiability to Solving PBESs

Verification Methodology:



Solving $\mathcal{E}$ answers $P \models \phi$

Transforming Satisfiability to Solving PBESs

Transformation of the First-order Modal μ-Calculus to PBES

First-order Modal μ-Calculus model checking problem

- Given is a First-order Modal μ-Calculus formula $\psi := \sigma Z(d_f : D_f := e_f).\ \phi$
- Given a system described by an LPE $X(e)$ over *Act*:

$$X(d{:}D) = \sum_{i \leq n} \sum_{e_i : D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$$

Compute whether $X(e) \models \psi$

- Transform the model checking problem to solving a PBES
- The transformation is similar to the transformation to BES.
- Idea: for each fixed point subformula $\sigma X(d_f : D_f := e_f).\ \chi$ of ψ, add an equation

$$(\sigma \tilde{X}(d : D, d_f : D_f, Par(Z, \psi)) = RHS(\chi))$$

- The order of the equations respects the subterm ordering in ψ

Transforming Satisfiability to Solving PBESs

Step 1: Create the outline of the PBES from the modal formula.

- Let $\psi := \sigma Z(d_f : D_f := e_f).\ \phi$
- Let $X(d : D) = \sum_{i \leq n} \sum_{e_i : D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$

Step 1

$$\mathbf{E}(\phi) = \epsilon \quad \text{for } \phi \in \{\text{true}, \text{false}, b, Z(e)\}$$

$$\mathbf{E}(\phi_1 \Box \phi_2) = \mathbf{E}(\phi_1)\ \mathbf{E}(\phi_2) \quad \text{for } \Box \in \{\vee, \wedge\}$$

$$\mathbf{E}(\mathbf{Q}d : D.\phi) = \mathbf{E}(\phi) \quad \text{for } \mathbf{Q} \in \{\exists, \forall\}$$

$$\mathbf{E}(\mathbf{M}_\alpha \phi) = \mathbf{E}(\phi) \quad \text{for } \mathbf{M}_\alpha \in \{[\alpha], \langle \alpha \rangle\}$$

$$\mathbf{E}((\sigma Z(d_f : D_f := e_f).\phi)) = (\sigma Z(d : D, d_f : D_f, Par(Z, \psi)) = \mathbf{RHS}(\phi))\ \mathbf{E}(\phi)$$

Transforming Satisfiability to Solving PBESs

Example

The one-place buffer system described by process B:

$$\begin{aligned} B(b : Bool, n : Nat) \quad &= \quad \sum_{m:Nat} \quad b \longrightarrow r(m) \cdot B(\mathsf{false}, m) \\ &+ \quad \neg b \longrightarrow s(n) \cdot B(\mathsf{true}, n) \end{aligned}$$

- Property ψ: if the input stream is constant, so is the output stream:

  $$\forall k : Nat.\ (\nu X.(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)))$$

- Transform ψ to a formula χ that starts with a dummy fixed point:

  $$\nu A.\ \forall k : Nat.\ (\nu X.(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)))$$

- We have: $Par(A, \chi) = []$ and $Par(X, \chi) = [k : Nat]$

Transforming Satisfiability to Solving PBESs

Example

Applying operator $\mathbf{E}$ on χ given B:

$$
\begin{aligned}
& \mathbf{E}(\chi) \\
= \quad & \mathbf{E}(\underline{\nu A.}\ \chi_1) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1))\ \mathbf{E}(\chi_1) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1))\ \mathbf{E}(\forall k : Nat.\ \chi_2) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1))\ \mathbf{E}(\forall k : Nat.\ \chi_2) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1))\ \mathbf{E}(\nu X.\chi_3) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1)) \\
& (\nu \tilde{X}(b : Bool, n : Nat, k : Nat) = \mathbf{RHS}(\chi_3))\ \mathbf{E}(\chi_3) \\
= \quad & \ldots \\
& (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1)) \\
& (\nu \tilde{X}(b : Bool, n : Nat, k : Nat) = \mathbf{RHS}(\chi_3))
\end{aligned}
$$

So, $\mathbf{E}(\chi)$ yields two equations.

Transforming Satisfiability to Solving PBESs

Step 2: Create the right-hand sides of the PBES equations.

- Let $\psi := \sigma Z(d_f : D_f := e_f).\ \phi$
- Let $X(d : D) = \sum_{i \leq n} \sum_{e_i : D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$

Step 2(1)

$$\mathbf{RHS}(\phi) = \phi \quad \text{for } \phi \in \{\text{true}, \text{false}, b\}$$

$$\mathbf{RHS}(Z(e)) = \tilde{Z}(d, e, Par(Z, \psi))$$

$$\mathbf{RHS}(\phi_1 \Box \phi_2) = \mathbf{RHS}(\phi_1) \Box \mathbf{RHS}(\phi_2) \quad \text{for } \Box \in \{\vee, \wedge\}$$

$$\mathbf{RHS}(\mathbf{Q}d : D.\phi) = \mathbf{Q}d : D.\ \mathbf{RHS}(\phi) \quad \text{for } \mathbf{Q} \in \{\exists, \forall\}$$

$$\mathbf{RHS}((\sigma Z(d_f : D_f := e_f).\phi)) = \tilde{Z}(d, e_f, Par(Z, \psi))$$

Transforming Satisfiability to Solving PBESs

Step 2: Create the right-hand sides of the PBES equations.

- Let $\psi := \sigma Z(d_f : D_f := e_f).\ \phi$
- Let $X(d : D) = \sum_{i \leq n} \sum_{e_i : D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$

Step 2(2)

$$\mathbf{RHS}([\alpha]\phi) = \bigwedge_{i \leq n} \forall e_i : D_i.\ \Big((c_i(d, e_i) \wedge \mathsf{match}(a_i(f_i(d, e_i)), \alpha)) \Rightarrow ((\mathbf{RHS}(\phi))[d := g_i(d, e_i)]) \Big)$$

$$\mathbf{RHS}(\langle\alpha\rangle\phi) = \bigvee_{i \leq n} \exists e_i : D_i.\ \Big(c_i(d, e_i) \wedge \mathsf{match}(a_i(f_i(d, e_i)), \alpha) \wedge ((\mathbf{RHS}(\phi))[d := g_i(d, e_i)]) \Big)$$

Transforming Satisfiability to Solving PBESs

Example

- Given the buffer process B of the previous example
- Consider subformula $(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X))$ of χ
- $$\begin{aligned} & \mathbf{RHS}(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)) \\ = \quad & \forall l : Nat.\ \mathbf{RHS}([r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)) \\ = \quad & \forall l : Nat.\ (\mathbf{RHS}([r(l)](l = k \Rightarrow X)) \wedge \mathbf{RHS}([s(l)](l = k \wedge X))) \end{aligned}$$

- Computing $\mathbf{RHS}([r(l)](l = k \Rightarrow X))$ requires process B.
- $$\begin{aligned} & \mathbf{RHS}(([r(l)](l = k \Rightarrow X))) \\ = \quad & (\forall m : Nat.\ (b \wedge \mathsf{match}(r(m), r(l))) \Rightarrow \mathbf{RHS}(l = k \Rightarrow X)[b := \mathsf{false}, n := m]) \\ \wedge \quad & ((\neg b \wedge \mathsf{match}(s(n), r(l))) \Rightarrow \mathbf{RHS}(l = k \Rightarrow X)[b := \mathsf{true}, n := n]) \\ = \quad & (\forall m : Nat.\ (b \wedge \mathsf{match}(r(m), r(l))) \Rightarrow (l = k \Rightarrow \tilde{X}(\mathsf{false}, m, k))) \\ \wedge \quad & ((\neg b \wedge \mathsf{match}(s(n), r(l))) \Rightarrow (l = k \Rightarrow \tilde{X}(\mathsf{true}, n, k))) \end{aligned}$$

Transforming Satisfiability to Solving PBESs

Matching parameterised actions with action formulae:

$$
\begin{array}{ll}
\mathsf{match}(a_i(d_{a_i}), \mathsf{true}) & = \mathsf{true} \\
\mathsf{match}(a_i(d_{a_i}), b) & = b \\
\mathsf{match}(a_i(d_{a_i}), a(e)) & = a \approx a_i \wedge d_{a_i} = e \\
\mathsf{match}(a_i(d_{a_i}), \neg\alpha) & = \neg\mathsf{match}(a_i(d_{a_i}), \alpha) \\
\mathsf{match}(a_i(d_{a_i}), \alpha_1 \wedge \alpha_2) & = \mathsf{match}(a_i(d_{a_i}), \alpha_1) \ \wedge \mathsf{match}(a_i(d_{a_i}), \alpha_2) \\
\mathsf{match}(a_i(d_{a_i}), \forall d : D.\alpha) & = \forall d : D.\mathsf{match}(a_i(d_{a_i}), \alpha)
\end{array}
$$

Observations:

- $\mathsf{match}(a_i(d_{a_i}))$ can always be brought into Positive Normal Form
- Hence, $\mathsf{match}(a_i(d_{a_i}))$ is a predicate formula
- $\mathsf{match}(a_i(d_{a_i}))$ does not introduce predicate variables

Transforming Satisfiability to Solving PBESs

Example

- The expression match$(r(m), r(l))$ yields $r = r \wedge m = l$, which simplifies to $m = l$
- The expression match$(s(n), r(l))$ yields $s = r \wedge n = l$, which simplifies to false
- An expression match$(s(n), \forall l : Nat.s(l))$ yields: $\forall l : Nat.$ match$(s(n), s(l))$, which yields $\forall l : Nat.\ (s = s \wedge n = l)$, which simplifies to $\forall l : Nat.\ n = l$, which further simplifies to false

Transforming Satisfiability to Solving PBESs

Example

Buffer system and constant stream revisited

$$\begin{aligned} B(b : Bool, n : Nat) \quad &= \quad \sum_{m:Nat} \; b \longrightarrow r(m) \cdot B(\mathsf{false}, m) \\ &+ \quad \neg b \longrightarrow s(n) \cdot B(\mathsf{true}, n) \end{aligned}$$

Property χ: $\nu A.\ \forall k : Nat.\ (\nu X.(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)))$

Result after translation to PBES $\mathcal{E}$ (note: cleanup using ordinary first-order logic):

$$(\nu \tilde{A}(b : Bool, n : Nat) = \forall k : Nat.\ \tilde{X}(b, n, k))$$

$$\begin{aligned} &(\nu \tilde{X}(b : Bool, n : Nat, k : Nat) = \\ &\quad \forall l : Nat.\ ((\forall m : Nat.\ (b \wedge m = l) \Rightarrow (l = k \Rightarrow \tilde{X}(\mathsf{false}, m, k))) \\ &\wedge ((\neg b \wedge n = l) \Rightarrow (l = k \wedge \tilde{X}(\mathsf{true}, n, k))))) \end{aligned}$$

For all $b : Bool$ and $n : Nat$, we have: $B(b, n) \models \psi$ iff $([\mathcal{E}]\theta\varepsilon)(\tilde{A})(b, n) = \mathsf{true}$

Outline

Solving PBESs

How to solve PBESs

$$X_i(e) \stackrel{?}{=} \text{true in } \mathcal{E} := \begin{array}{l} (\sigma_1 X_1(d_1 : D_1) = \phi_1) \\ \vdots \\ (\sigma_n X_n(d_n : D_n) = \phi_n) \end{array}$$

Known techniques for solving/simplifying $\mathcal{E}$:

- Instantiation to BES and subsequently solve the BES
- Gauß Elimination on PBES + symbolic approximation of equations
- Using patterns
- Using under/over approximation
- Invariants

Solving PBESs

Instantiation to BES:

$$X_i(e) \stackrel{?}{=} \text{true in } \mathcal{E} := \begin{array}{l} (\sigma_1 X_1(d_1 : D_1) = \phi_1) \\ \vdots \\ (\sigma_n X_n(d_n : D_n) = \phi_n) \end{array}$$

- Let X_i^e be a fresh propositional variable representing instance $X_i(e)$.
- The procedure below creates a BES from $\mathcal{E}$ s.t. $X_i(e) = \text{true}$ iff $X_i^e = \text{true}$
  1. For each $X_j(e_j)$ occurring in $\text{eval}(\phi_i[d_i := e])$ create a fresh variable $X_j^{e_j}$
  2. Create an equation $\sigma_i X_i^e = \tilde{\phi}_i$, where:
     - $\overline{\phi_i} = \text{eval}(\phi_i[d_i := e])$,
     - $\tilde{\phi}_i$ is $\overline{\phi_i}$ in which every $X_j(e_j)$ is replaced by $X_j^{e_j}$
  3. Repeat step 1 and 2 for every $X_j^{e_j}$ introduced in step 1
  4. Order all equations $\sigma_i X_i^e = \ldots$ according to the ordering of $\mathcal{E}$

Solving PBESs

Example

PBES: $(\nu X(n : Nat) = n \leq 2 \wedge Y(n))\ (\mu Y(n : Nat) = \mathsf{odd}(n) \vee X(n+1))$

Instantiation starting at e.g. $X(0)$ ..... introduce X^0

1. $Y(0)$ occurs in $\mathsf{eval}((n \leq 2 \wedge Y(n))[n := 0])$ ..... introduce Y^0
2. Introduce $\nu X^0 = \mathsf{eval}(0 \leq 2 \wedge Y^0)$ ..... $\nu X^0 = Y^0$
3. $X(1)$ occurs in $\mathsf{eval}((\mathsf{odd}(n) \vee X(n+1))[n := 0])$ ..... introduce X^1
4. Introduce $\mu Y^0 = \mathsf{eval}(\mathsf{odd}(0) \vee X^1)$ ..... $\mu Y^0 = X^1$
5. $Y(1)$ occurs in $\mathsf{eval}((n \leq 2 \wedge Y(n))[n := 1])$ ..... introduce Y^1
6. Introduce $\nu X^1 = \mathsf{eval}(1 \leq 2 \wedge Y^1)$ ..... $\nu X^1 = Y^1$
7. no variable occurs in $\mathsf{eval}((\mathsf{odd}(n) \vee X(n+1))[n := 1])$ ..... end
8. Introduce $\mu Y^1 = \mathsf{eval}(\mathsf{odd}(1) \vee X^2)$ ..... $\mu Y^1 = \mathsf{true}$
9. Order equations: first X^i, then Y^j
10. Resulting BES: $(\nu X^0 = Y^0)\ (\nu X^1 = Y^1)\ (\mu Y_0 = X^1)\ (\mu Y_1 = \mathsf{true})$

Solving PBESs

Definition (Logical Equivalence)

Let ϕ, ψ be two predicates. Then ψ is logically equivalent to ϕ, denoted $\phi \leftrightarrow \psi$ iff

$$\forall \varepsilon, \eta : \ [\phi]\eta\varepsilon = [\psi]\eta\varepsilon$$

- If $\phi \leftrightarrow \psi$, then equation $\nu X(d : D) = \phi$ has the same solution as $\nu X(d : D) = \psi$ (likewise for μ)
- Useful simplifications:
  - false $\wedge\ \phi \leftrightarrow$ false
  - true $\vee\ \phi \leftrightarrow$ true
  - if $d \notin \mathrm{FV}(\phi)$, then $(\exists d : D.\ \phi) \leftrightarrow (\forall d : D.\ \phi) \leftrightarrow \phi$
  - One-point rule: $(\exists d : D. d = e \wedge \phi(d)) \leftrightarrow \phi(e)$
  - One-point rule: $(\forall d : D. d = e \Rightarrow \phi(d)) \leftrightarrow \phi(e)$
- Apply logical simplifications before applying any other PBES manipulations.

Outline

PBES Manipulation

Instantiation may not terminate

- Consider the below formula that asserts that an infinite a-path is possible:

  $$\nu X.\langle a \rangle X$$

- Consider the infinite state process M:

  $$M(n : Nat) = \mathsf{true} \longrightarrow a \cdot M(n+1)$$

- Resulting PBES:

  $$\nu \tilde{X}(n : Nat) = \mathsf{true} \wedge \tilde{X}(n+1)$$

- Problem: always one new variable must be further investigated:
  X^0 depends on X^1 depends on X^2 depends on...

PBES Manipulation

Definition (Simple Formula)

A simple formula is a formula not containing predicate variables

Observations:

1. Consider the equation $\nu\tilde{X}(n : Nat) = \text{true} \wedge \tilde{X}(n+1)$
   - $\tilde{X}$ has solution true (check!)
   - Consider formal parameter n:
   - It does not affect the value of the simple subformula true
   - It appears to be redundant for the solution to $\tilde{X}$
2. Consider the equation $\nu\tilde{X}(n : Nat, m : Nat) = n \leq 5 \wedge \tilde{X}(n+m, m)$
   - $\tilde{X}$ has solution $n \leq 5 \wedge m = 0$ (check!)
   - Consider formal parameter m:
   - It does not affect the value of the simple formula $n \leq 5$
   - Via a single recursion through $\tilde{X}$, it does affect the value of $n \leq 5$
   - It appears to become significant for the solution to $\tilde{X}$

PBES Manipulation

Removing redundancy from a PBES (step 1):

- Identify all obvious significant formal parameters:

$$
\begin{array}{ll}
\mathsf{sig}(b) & = \mathsf{FV}(b) \\
\mathsf{sig}(\tilde{X}(e)) & = \emptyset \\
\mathsf{sig}(\phi_1 \wedge \phi_2) & = \mathsf{sig}(\phi_1) \cup \mathsf{sig}(\phi_2) \\
\mathsf{sig}(\phi_1 \vee \phi_2) & = \mathsf{sig}(\phi_1) \cup \mathsf{sig}(\phi_2) \\
\mathsf{sig}(\forall d{:}D.\ \phi) & = \mathsf{sig}(\phi) \setminus \{d\} \\
\mathsf{sig}(\exists d{:}D.\ \phi) & = \mathsf{sig}(\phi) \setminus \{d\}
\end{array}
$$

- Example: $\mathsf{sig}(\mathsf{true} \wedge \tilde{X}(n+1)) = \emptyset$
- Example: $\mathsf{sig}(n \leq 5 \wedge \tilde{X}(n+m, m)) = \{n\}$

PBES Manipulation

Removing redundancy from a PBES (step 2):

- Identify the dependencies:

$$
\begin{array}{ll}
\mathsf{dep}(b) & = \emptyset \\
\mathsf{dep}(\tilde{X}(e)) & = \{X(e)\} \\
\mathsf{dep}(\phi_1 \wedge \phi_2) & = \mathsf{dep}(\phi_1) \cup \mathsf{dep}(\phi_2) \\
\mathsf{dep}(\phi_1 \vee \phi_2) & = \mathsf{dep}(\phi_1) \cup \mathsf{dep}(\phi_2) \\
\mathsf{dep}(\forall d{:}D.\ \phi) & = \mathsf{dep}(\phi) \\
\mathsf{dep}(\exists d{:}D.\ \phi) & = \mathsf{dep}(\phi)
\end{array}
$$

- Example: $\mathsf{dep}(\mathsf{true} \wedge \tilde{X}(n+1)) = \{X(n+1)\}$
- Example: $\mathsf{dep}(n \leq 5 \wedge \tilde{X}(n+m, m)) = \{X(n+m, m)\}$

PBES Manipulation

Removing redundancy from a PBES (step 3):

Assume the following PBES:

$$\begin{aligned}\mathcal{E} := \quad & (\sigma_1 X_1(d_1 : D_1) = \phi_1) \\ & \vdots \\ & (\sigma_n X_n(d_n : D_n) = \phi_n)\end{aligned}$$

- $\text{arity}(X_i)$: the length of vector d_i
- $d_i[j]$ denotes the j-th element of vector d_i

- Construct a marked influence graph $G(\mathcal{E}) = \langle V, \longrightarrow, M \rangle$:
- $V = \{(X_i, j) \mid 1 \leq j \leq \text{arity}(X_i)\}$ is the set of vertices
- $(X_i, k) \longrightarrow (X_j, l)$ iff for some expression e: $X_j(e) \in \text{dep}(\phi_i)$ and $d_i[k] \in \text{FV}(e[l])$
- $M = \{(X_i, j) \mid 1 \leq i \leq n \text{ and } d_i[j] \in \text{sig}(\phi_i)\}$ is the marking

PBES Manipulation

Definition (Positive redundant parameters)

Given a Marked Influence Graph $G(\mathcal{E}) = \langle V, \longrightarrow, M \rangle$.
The set of positive redundant parameters of $\mathcal{E}$ is:

$$\mathcal{R} = \{d_i[j] \mid (X_i, j) \not\longrightarrow^* (X_k, l) \text{ and } (X_k, l) \in M\}$$

- Computing the set $\mathcal{R}$ requires $\mathcal{O}(| \longrightarrow |)$ steps at most
- $\mathcal{R}$ can be computed using a standard least fixed point computation, a depth-first search or a breadth-first search.

PBES Manipulation

- Let $\mathcal{E}$ be a closed equation system with no unbound data variables
- Let $\mathcal{R}$ be the set of positive redundant parameters associated to $\mathcal{E}$
- The equation system $\widehat{\mathcal{E}}$ is obtained from $\mathcal{E}$ as follows:
- remove a parameter $d_i[j]$ from $X_i(d_i{:}D_i)$ iff $d_i[j] \in \mathcal{R}$
- remove an expression $e[j]$ from an occurrence $X_i(e)$ iff $d_i[j] \in \mathcal{R}$

Theorem (Redundancy)

For closed $\mathcal{E}$ that contains no unbound data variables, $\mathcal{E}$ and $\widehat{\mathcal{E}}$ have the "same" solutions, i.e., the solution of a variable X does not depend on the parameters that have been identified as positively redundant.

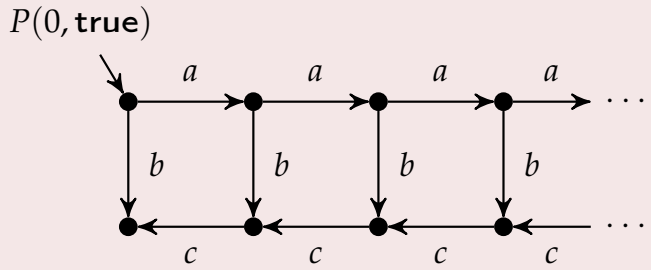
Note: typing of predicate variables in $\mathcal{E}$ can differ from those in $\widehat{\mathcal{E}}$.

PBES Manipulation

Example

Consider the following process:

$$
\begin{aligned}
& P(n:Nat, d:Bool) \\
= \quad & d \longrightarrow a \cdot P(n+1, d) \\
+ \quad & d \longrightarrow b \cdot P(n, \neg d) \\
+ \quad & \neg d \wedge n > 0 \longrightarrow c \cdot P(n-1, d)
\end{aligned}
$$



Along every a path, always a b action is attainable:

$$\nu V.\ ([a]V \wedge \mu W.\ (\langle a \rangle W \vee \langle b \rangle \mathsf{true}))$$

PBES:
$$
\begin{aligned}
& \big(\nu V(n:Nat, d:Bool) = \quad (d \implies V(n+1, d)) \wedge W(n, d)\big) \\
& \big(\mu W(n:Nat, d:Bool) = \quad d \vee (d \wedge W(n+1, d))\big)
\end{aligned}
$$

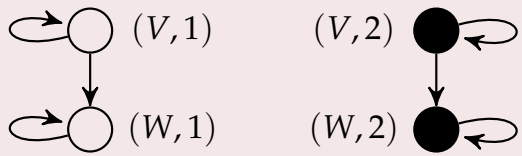
Instantiation of the PBES does not terminate.

PBES Manipulation

Example (Cont'd)

PBES: $\begin{array}{ll} (\nu V(n:Nat, d:Bool) = & (d \implies V(n+1,d)) \wedge W(n,d)) \\ (\mu W(n:Nat, d:Bool) = & d \vee (d \wedge W(n+1,d))) \end{array}$

- $\mathrm{dep}(d \implies V(n+1,d)) \wedge W(n,d)) = \{V(n+1,d), W(n,d)\}$
- $\mathrm{dep}(d \vee (d \wedge W(n+1,d))) = \{W(n+1,d)\}$
- Marked Influence Graph ($\mathrm{arity}(V) = \mathrm{arity}(W) = 2$):



Marked states are black; non-marked white

- $\mathcal{R} = \{(V,1), W(1)\}$, i.e., parameter n is positively redundant for V and W.
- Reduced PBES: $\begin{array}{ll} (\nu V(d:Bool) = & (d \implies V(d)) \wedge W(d)) \\ (\mu W(d:Bool) = & d \vee (d \wedge W(d))) \end{array}$
- Instantiation of the above PBES terminates

Outline

Exercise

Consider the lossy channel system described by the following LPE:

$$\begin{aligned} C(b:Bool, m:M) \quad &= \quad \sum_{k:M} \; b \longrightarrow r(k) \cdot C(\mathsf{false}, k) \\ &+ \quad \neg b \longrightarrow s(m) \cdot C(\mathsf{true}, m) \\ &+ \quad \neg b \longrightarrow l \cdot C(\mathsf{true}, m) \end{aligned}$$

Action r stands for reading, s stands for sending and l stands for losing a message.

1. $\nu X.([\mathsf{true}]X \wedge (\mu Y.[\neg(\exists m:M.s(m))]Y \wedge \langle \mathsf{true} \rangle \mathsf{true}))$
2. $\nu X.\mu Y.\nu Z.[\exists m{:}M.s(m)]X \wedge ([\exists m{:}M.s(m)]\mathsf{false} \vee [\neg(\exists m{:}M.s(m))]Y) \wedge [\neg(\exists m{:}M.s(m))]Z$

Questions:

- Translate both formulae to PBESs given process $C(\mathsf{true}, m_0)$
- Use instantiation to compute BESs when $M = Bool$, and solve the BES ($m_0 = \mathsf{true}$)
- Can you remove redundant parameters? If so, remove these redundant parameters and try instantiation to compute a BES when $M = Nat$ ($m_0 = 0$)