Algorithms for Model Checking (2IW55)

Lecture 10

Parameterised Boolean Equation Systems (2)

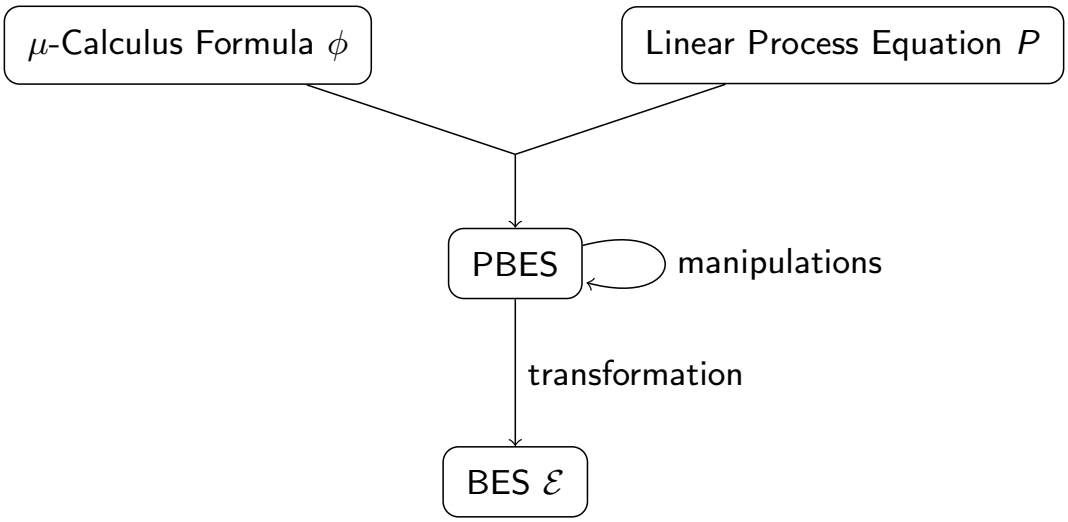
Background material:

- *Verification of Reactive Systems via Instantiation of Parameterised Boolean Equation Systems*, B. Ploeger, J.W. Wesselink and T.A.C. Willemse (*I&C 2010/2011*)
- *Static Analysis Techniques for Parameterised Boolean Equation Systems*, S. Orzan, J.W. Wesselink and T.A.C. Willemse (*TACAS 2009*)

Tim Willemse
(timw@win.tue.nl)
http://www.win.tue.nl/∼timw
HG 6.76

Verification via PBESs

Verification Methodology:



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

First-order Modal μ-Calculus model checking problem

- Given is a First-order Modal μ-Calculus formula $\sigma Z.\ \phi$
- Given a system described by an LPE $X(e)$

Compute whether $X(e) \models \sigma Z.\ \phi$

- Transform the model checking problem to solving a PBES
- The transformation is similar to the transformation to BES.
- Idea: for each fixed point subformula $\sigma' X.\ \psi$ of $\sigma Z.\phi$, add an equation

$$\sigma' \tilde{X}(d : D, \cdots) = RHS(\psi)$$

- The order of the equations respects the subterm ordering in $\sigma Z.\phi$

- Identify a list of data variables bound outside the scope of a fixed point formula
- Given a formula Φ and some formal variable Z

Identify Bound Data Variables

$$Par(Z, b, l) = Par(Z, X, l) = []$$

$$Par(Z, \phi \wedge \psi, l) = Par(Z, \phi \vee \psi, l) = Par(Z, \phi, l) +\!\!+ Par(Z, \psi, l)$$

$$Par(Z, \forall d{:}D.\phi, l) = Par(Z, \exists d{:}D.\phi, l) = Par(Z, \phi, [d{:}D] +\!\!+ l)$$

$$Par(Z, [\alpha]\phi, l) = Par(Z, \langle\alpha\rangle\phi, l) = Par(Z, \phi, l)$$

$$Par(Z, \sigma X.\phi, l) = \begin{cases} l & \text{if } Z = X \\ Par(Z, \phi, l) & \text{otherwise} \end{cases}$$

Example

The one-place buffer system described by process B:

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

- Property Φ: 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 Φ to a formula Ψ 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: $Par(A, \Psi, []) = []$ and $Par(X, \Psi, []) = [k : Nat]$

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

Create Equation System Outline

$$\begin{aligned} \mathbf{E}(b) &= \epsilon & \mathbf{E}(Z) &= \epsilon \\ \mathbf{E}(\phi \wedge \psi) &= \mathbf{E}(\phi)\ \mathbf{E}(\psi) & \mathbf{E}(\phi \vee \psi) &= \mathbf{E}(\phi)\ \mathbf{E}(\psi) \\ \mathbf{E}(\forall d'{:}D'.\phi) &= \mathbf{E}(\phi) & \mathbf{E}(\exists d'{:}D'.\phi) &= \mathbf{E}(\phi) \\ \mathbf{E}([\alpha]\phi) &= \mathbf{E}(\phi) & \mathbf{E}(\langle\alpha\rangle\phi) &= \mathbf{E}(\phi) \end{aligned}$$

$$\mathbf{E}(\sigma Z.\phi) = \Big(\sigma Z(d{:}D, Par(Z, \psi, [])) = \mathbf{RHS}(\phi)\Big)\ \mathbf{E}(\phi)$$

Example

Applying operator $\mathbf{E}$ on formula Ψ given the buffer process B:

$$
\begin{aligned}
& \mathbf{E}(\Psi) \\
= \quad & \mathbf{E}(\underline{\nu A.}\ \Psi_1) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\Psi_1))\ \mathbf{E}(\Psi_1) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\Psi_1))\ \mathbf{E}(\forall k : Nat.\ \Psi_2) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\Psi_1))\ \mathbf{E}(\forall k : Nat.\ \Psi_2) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\Psi_1))\ \mathbf{E}(\nu X.\Psi_3) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\Psi_1)) \\
& (\nu \tilde{X}(b : Bool, n : Nat, k : Nat) = \mathbf{RHS}(\Psi_3))\ \mathbf{E}(\Psi_3) \\
= \quad & \ldots \\
& (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\Psi_1)) \\
& (\nu \tilde{X}(b : Bool, n : Nat, k : Nat) = \mathbf{RHS}(\Psi_3))
\end{aligned}
$$

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

- Let $\Phi := \sigma Y.\ \phi$
- Given LPE $X(d{:}D) = \sum_{i \le n} \sum_{e_i : D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$

RHS:

$$
\begin{aligned}
&\mathbf{RHS}(b) && = b && \qquad &\mathbf{RHS}(Z) && = \tilde{Z}(d, Par(Z, \Phi, [])) \\
&\mathbf{RHS}(\phi \wedge \psi) && = \mathbf{RHS}(\phi)\ \wedge\ \mathbf{RHS}(\psi) && \qquad &\mathbf{RHS}(\phi \vee \psi) && = \mathbf{RHS}(\phi)\ \vee\ \mathbf{RHS}(\psi) \\
&\mathbf{RHS}(\forall d'{:}D'.\phi) && = \forall d'{:}D'.\ \mathbf{RHS}(\phi) && \qquad &\mathbf{RHS}(\exists d'{:}D'.\phi) && = \exists d'{:}D'.\ \mathbf{RHS}(\phi) \\
&\mathbf{RHS}(\sigma Z.\phi) && = \tilde{Z}(d, Par(Z, \Phi, []))
\end{aligned}
$$

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

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

Example (Verification of the Buffer process B, continued)

- Consider subformula $(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X))$ of Ψ

$$\begin{aligned} & \mathbf{RHS}(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)) \\ = \quad & \forall l : Nat.\ \mathbf{RHS}([r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)) \\ = \quad & \forall l : Nat.\ (\mathbf{RHS}([r(l)](l = k \Rightarrow X)) \wedge \mathbf{RHS}([s(l)](l = k \wedge X))) \end{aligned}$$

- Computing $\mathbf{RHS}([r(l)](l = k \Rightarrow X))$ requires process B.

$$\begin{aligned} & \mathbf{RHS}(([r(l)](l = k \Rightarrow X))) \\ = \quad & (\forall m : Nat.\ (b \wedge r(m) \text{ in } r(l)) \Rightarrow \mathbf{RHS}(l = k \Rightarrow X)[b := \text{false}, n := m]) \\ \wedge \quad & ((\neg b \wedge s(n) \text{ in } r(l)) \Rightarrow \mathbf{RHS}(l = k \Rightarrow X)[b := \text{true}, n := n]) \\ = \quad & (\forall m : Nat.\ (b \wedge r(m) \text{ in } r(l)) \Rightarrow (l = k \Rightarrow \tilde{X}(\text{false}, m, k))) \\ \wedge \quad & ((\neg b \wedge s(n) \text{ in } r(l)) \Rightarrow (l = k \Rightarrow \tilde{X}(\text{true}, n, k))) \end{aligned}$$

Matching parameterised actions with action formulae:

$$\begin{aligned} a(e) \text{ in } \text{true} &= \text{true} \\ a(e) \text{ in } a'(e') &= (a = a' \wedge e = e') \\ a(e) \text{ in } \neg\alpha &= \neg(a(e) \text{ in } \alpha) \\ a(e) \text{ in } (\alpha \wedge \beta) &= (a(e) \text{ in } \alpha) \ \wedge (a(e) \text{ in } \beta) \\ a(e) \text{ in } (\alpha \vee \beta) &= (a(e) \text{ in } \alpha) \ \vee (a(e) \text{ in } \beta) \end{aligned}$$

Observations:

- in yields a predicate formula
- in does not introduce predicate variables

Example

- The expression $r(m)$ in $r(l)$ yields $r = r \wedge m = l$, which simplifies to $m = l$
- The expression $s(n)$ in $r(l)$ yields $s = r \wedge n = l$, which simplifies to false

Example (Verification of the Buffer process, continued)

Buffer system and constant stream revisited

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

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

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

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

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

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

How to solve PBESs

$$X_i(e) \stackrel{?}{=} \text{true in } \mathcal{E} := \quad (\sigma_1 X_1(d_1 : D_1) = \phi_1) \cdots (\sigma_n X_n(d_n : D_n) = \phi_n)$$

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

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 PBES manipulations/solving techniques.

Instantiation to BES:

$$X_i(e) \stackrel{?}{=} \text{true in } \mathcal{E} := \quad (\sigma_1 X_1(d_1 : D_1) = \phi_1) \cdots (\sigma_n X_n(d_n : D_n) = \phi_n)$$

- 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, for which there is no equation
  4. Order all equations $\sigma_i X_i^e = \ldots$ according to the ordering of $\mathcal{E}$ (ordering within a block may be arbitrary)

Example

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

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

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

Gauß elimination on PBESs + Symbolic Approximation:

$X_i(e) \stackrel{?}{=}$ true in $\mathcal{E} := \quad (\sigma_1 X_1(d_1 : D_1) = \phi_1) \cdots (\sigma_n X_n(d_n : D_n) = \phi_n)$

- Local solution: eliminate X in its defining equation:

$$\mathcal{E}_0\ (\sigma X(d{:}D) = \phi)\ \mathcal{E}_1 \text{ becomes } \mathcal{E}_0\ (\sigma X(d{:}D) = X^\omega)\ \mathcal{E}_1$$

  - X^ω can be found by symbolic approximation:
  - $X^0 =$ false if $\sigma = \mu$, else $X^0 =$ true
  - $X^{n+1} = \phi[X := X^n]$
  - X^ω may require transfinite approximation; else $X^\omega = X^n$ for $X^n \leftrightarrow X^{n+1}$
- Substitute solved equations (i.e. not containing predicate variables) forward:

$$\begin{array}{rl} & \mathcal{E}_0\ (\sigma_1 X_1(d_1{:}D_1) = \phi_1)\ \mathcal{E}_1\ (\sigma_2 X_2(d_2{:}D_2) = \phi_2)\ \mathcal{E}_2 \\ \text{becomes:} & \mathcal{E}_0\ (\sigma_1 X_1(d_1{:}D_1) = \phi_1)\ \mathcal{E}_1\ (\sigma_2 X_2(d_2{:}D_2) = \phi_2[X_1 := \phi_1])\ \mathcal{E}_2 \end{array}$$

- Substitute definition backwards:

$$\begin{array}{rl} & \mathcal{E}_0\ (\sigma_1 X_1(d_1{:}D_1) = \phi_1)\ \mathcal{E}_1\ (\sigma_2 X_2(d_2{:}D_2) = \phi_2)\ \mathcal{E}_2 \\ \text{becomes:} & \mathcal{E}_0\ (\sigma_1 X_1(d_1{:}D_1) = \phi_1[X_2 := \phi_2])\ \mathcal{E}_1\ (\sigma_2 X_2(d_2{:}D_2) = \phi_2)\ \mathcal{E}_2 \end{array}$$

Example

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

1. Eliminate Y from $(\mu Y(n : Nat) = \text{odd}(n) \vee X(n+1))$ ..... done
2. Substitute definition of Y backwards:

$$\begin{array}{rl} & (\nu X(n : Nat) = n \leq 2 \wedge Y(n)) \\ \text{becomes} & (\nu X(n : Nat) = n \leq 2 \wedge (\text{odd}(n) \vee X(n+1))) \end{array}$$

3. Eliminate X from $(\nu X(n : Nat) = n \leq 2 \wedge (\text{odd}(n) \vee X(n+1)))$:

$$\begin{array}{lll} X^0 & \equiv \text{true} & \\ X^1 & \equiv n \leq 2 \wedge (\text{odd}(n) \vee \text{true}) & \equiv n \leq 2 \\ X^2 & \equiv n \leq 2 \wedge (\text{odd}(n) \vee n+1 \leq 2) & \equiv n \leq 2 \wedge (\text{odd}(n) \vee n \leq 1) \quad \equiv n \leq 1 \\ X^3 & \equiv n \leq 2 \wedge (\text{odd}(n) \vee n+1 \leq 1) & \equiv n \leq 2 \wedge (\text{odd}(n) \vee n = 0) \quad \equiv n \leq 1 \end{array}$$

So, solution to X is $n \leq 1$

Solving PBESs

Gauß Elimination terminates; symbolic approximation may not terminate

- Due to infinite data types, a transfinite approximation may be needed
- Evaluating predicates may be impossible: $\exists k, l, m : Nat.x^k + y^l = z^m$
- Theorem proving technology may be added in symbolic approximation

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:

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

- Resulting PBES:

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

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

Definition (Simple Formula)

A simple formula is a formula not containing predicate variables

Observations:

1. Consider the equation $\nu X(n : Nat) = \text{true} \wedge X(n+1)$
   - 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 X
2. Consider the equation $\nu X(n : Nat, m : Nat) = n \leq 5 \wedge X(n+m, m)$
   - 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 X, it does affect the value of $n \leq 5$
   - It appears to become significant for the solution to X

- Identify all obvious significant formal parameters . . . . . sig
- Identify the dependencies . . . . . dep

$$
\begin{array}{ll}
\mathrm{sig}(b) & = \mathrm{FV}(b) \\
\mathrm{sig}(X(e)) & = \emptyset \\
\mathrm{sig}(\phi \wedge \psi) & = \mathrm{sig}(\phi) \cup \mathrm{sig}(\psi) \\
\mathrm{sig}(\phi \vee \psi) & = \mathrm{sig}(\phi) \cup \mathrm{sig}(\psi) \\
\mathrm{sig}(\forall d{:}D.\ \phi) & = \mathrm{sig}(\phi) \setminus \{d\} \\
\mathrm{sig}(\exists d{:}D.\ \phi) & = \mathrm{sig}(\phi) \setminus \{d\}
\end{array}
\qquad
\begin{array}{ll}
\mathrm{dep}(b) & = \emptyset \\
\mathrm{dep}(X(e)) & = \{X(e)\} \\
\mathrm{dep}(\phi \wedge \psi) & = \mathrm{dep}(\phi) \cup \mathrm{dep}(\psi) \\
\mathrm{dep}(\phi \vee \psi) & = \mathrm{dep}(\phi) \cup \mathrm{dep}(\psi) \\
\mathrm{dep}(\forall d{:}D.\ \phi) & = \mathrm{dep}(\phi) \\
\mathrm{dep}(\exists d{:}D.\ \phi) & = \mathrm{dep}(\phi)
\end{array}
$$

Examples:

- $\mathrm{sig}(\text{true} \wedge X(n+1)) = \emptyset$, $\mathrm{sig}(n \leq 5 \wedge X(n+m, m)) = \{n\}$
- $\mathrm{dep}(\text{true} \wedge X(n+1)) = \{X(n+1)\}$, $\mathrm{dep}(n \leq 5 \wedge X(n+m, m)) = \{X(n+m, m)\}$

Assume the following PBES:

$$\mathcal{E} := \quad (\sigma_1 X_1(d_1 : D_1) = \phi_1) \cdots (\sigma_n X_n(d_n : D_n) = \phi_n)$$

- 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

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

Given closed equation system $\mathcal{E}$ with no unbound data variables

Procedure for eliminating redundant parameters in $\mathcal{E}$

1. Step 1 (compute redundant parameters)
   - 1.1 Construct Marked Influence Graph of $\mathcal{E}$
   - 1.2 Compute the set $\mathcal{R}$ of positive redundant parameters of $\mathcal{E}$
2. Step 2 (remove redundant parameters): for every equation $\sigma_i X_i(d_i{:}D_i) = \phi_i$ in $\mathcal{E}$:
   - 2.1 remove parameter $d_i[j]$ from $X_i(d_i{:}D_i)$ iff $d_i[j] \in \mathcal{R}$
   - 2.2 remove expression $e[j]$ from an occurrence $X_k(e)$ in ϕ_i iff $d_k[j] \in \mathcal{R}$

Theorem (Redundancy)

The modified equation system $\mathcal{E}$ has the "same" solution as $\mathcal{E}$, i.e., the solution of a variable X does not depend on the parameters that have been identified as positively redundant.

Example

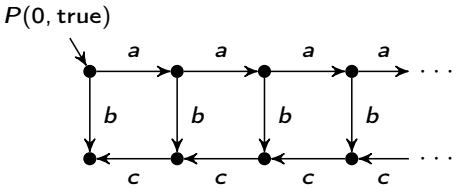
- $\nu X(b{:}Bool, n{:}Nat) = b \wedge X(b, n+1)$ has solution $f := \lambda(c, v){\in}Bool \times Nat.\ c$
- $\nu X(b{:}Bool) = b \wedge X(b)$ has solution $g := \lambda c{\in}Bool.\ c$
- For all $c{\in}Bool$, $v{\in}Nat$, $f(c, v) = g(c)$.

PBES Manipulation

Example

Consider the following process:

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



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

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

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

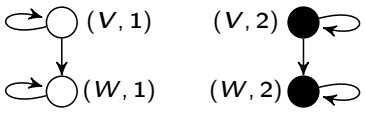
Instantiation of the PBES does not terminate.

PBES Manipulation

Example (Cont'd)

PBES: $(\nu V(n:Nat, d:Bool) = \quad (d \implies V(n+1, d)) \wedge W(n, d))$
$(\mu W(n:Nat, d:Bool) = \quad d \vee (d \wedge W(n+1, d)))$

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



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: $(\nu V(d:Bool) = \quad (d \implies V(d)) \wedge W(d))$
$(\mu W(d:Bool) = \quad d \vee (d \wedge W(d)))$
- Instantiation of the above PBES terminates

Exercise

Consider the lossy channel system described by the following LPE:

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

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

1. $\nu X.([\text{true}]X \wedge (\mu Y.[l]Y \wedge \forall m{:}M.[r(m)]Y \wedge \langle\text{true}\rangle\text{true}))$
2. $\nu X.\mu Y.\nu Z.(\forall m{:}M.[s(m)]X) \wedge ((\forall m{:}M.\ [s(m)]\text{false}) \vee ([l]Y \wedge \forall m{:}M.[r(m)]Y)) \wedge [l]Z \wedge \forall m{:}M.[r(m)]Z$

Questions:

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