

Algorithms for Model Checking (2IW55)

Lecture 15

Parameterised Boolean Equation Systems

Tim Willemse

(timw@win.tue.nl)

<http://www.win.tue.nl/~timw>

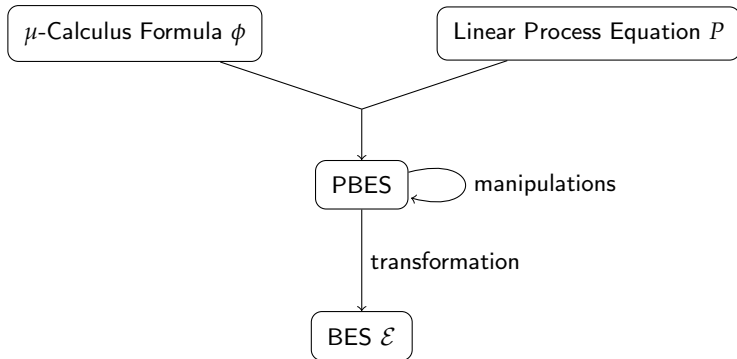
HG 6.81

Outline

- 1 Transforming Satisfiability to Solving PBESs
- 2 Solving PBESs
- 3 PBES Manipulation
- 4 Exercise

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

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

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

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

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

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

Transforming Satisfiability to Solving PBESs

Example

The one-place buffer system described by process B :

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

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

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

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

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

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

Transforming Satisfiability to Solving PBESs

Example

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

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

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)

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

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

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

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

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

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)

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

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

Transforming Satisfiability to Solving PBESs

Example

- Given the buffer process B of the previous example
- Consider subformula $(\forall l : \text{Nat}. [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X))$ of χ
- $$\begin{aligned} & \mathbf{RHS}(\forall l : \text{Nat}. [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)) \\ &= \forall l : \text{Nat}. \mathbf{RHS}([r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)) \\ &= \forall l : \text{Nat}. (\mathbf{RHS}([r(l)](l = k \Rightarrow X)) \wedge \mathbf{RHS}([s(l)](l = k \wedge X))) \end{aligned}$$
- Computing $\mathbf{RHS}([r(l)](l = k \Rightarrow X))$ requires process B .
- $$\begin{aligned} & \mathbf{RHS}([r(l)](l = k \Rightarrow X)) \\ &= (\forall m : \text{Nat}. (b \wedge \text{match}(r(m), r(l))) \Rightarrow \mathbf{RHS}(l = k \Rightarrow X)[b := \text{false}, n := m]) \\ & \wedge ((\neg b \wedge \text{match}(s(n), r(l))) \Rightarrow \mathbf{RHS}(l = k \Rightarrow X)[b := \text{true}, n := n]) \\ &= (\forall m : \text{Nat}. (b \wedge \text{match}(r(m), r(l))) \Rightarrow (l = k \Rightarrow \tilde{X}(\text{false}, m, k))) \\ & \wedge ((\neg b \wedge \text{match}(s(n), r(l))) \Rightarrow (l = k \Rightarrow \tilde{X}(\text{true}, n, k))) \end{aligned}$$

Transforming Satisfiability to Solving PBESs

Matching parameterised actions with action formulae:

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

Observations:

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

Transforming Satisfiability to Solving PBESs

Example

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

Transforming Satisfiability to Solving PBESs

Example

Buffer system and constant stream revisited

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

Property 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):

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

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

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

Outline

- 1 Transforming Satisfiability to Solving PBESs
- 2 Solving PBESs
- 3 PBES Manipulation
- 4 Exercise

Solving PBESs

How to solve PBESs

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

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:

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

- 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 $\sigma_i X_i^e = \bar{\phi}_i$, where:
 - $\bar{\phi}_i = \text{eval}(\phi_i[d_i := e])$,
 - $\bar{\phi}_i$ is ϕ_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 $\sigma_i X_i^e = \dots$ according to the ordering of \mathcal{E}

Solving PBESs

Example

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

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: $(\nu X^0 = Y^0)$ $(\nu X^1 = Y^1)$ $(\mu Y_0 = X^1)$ $(\mu Y_1 = \text{true})$

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:
 - $\text{false} \wedge \phi \leftrightarrow \text{false}$
 - $\text{true} \vee \phi \leftrightarrow \text{true}$
 - if $d \notin \text{FV}(\phi)$, 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

- 1 Transforming Satisfiability to Solving PBESs
- 2 Solving PBESs
- 3 PBES Manipulation**
- 4 Exercise

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 :

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

- Resulting PBES:

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

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

PBES Manipulation

Definition (Simple Formula)

A **simple formula** is a formula not containing predicate variables

Observations:

- 1 Consider the equation $v\tilde{X}(n : \text{Nat}) = \text{true} \wedge \tilde{X}(n + 1)$
 - \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 $v\tilde{X}(n : \text{Nat}, m : \text{Nat}) = n \leq 5 \wedge \tilde{X}(n + m, m)$
 - \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:

$$\text{sig}(b) = \text{FV}(b)$$

$$\text{sig}(\tilde{X}(e)) = \emptyset$$

$$\text{sig}(\phi_1 \wedge \phi_2) = \text{sig}(\phi_1) \cup \text{sig}(\phi_2)$$

$$\text{sig}(\phi_1 \vee \phi_2) = \text{sig}(\phi_1) \cup \text{sig}(\phi_2)$$

$$\text{sig}(\forall d:D. \phi) = \text{sig}(\phi) \setminus \{d\}$$

$$\text{sig}(\exists d:D. \phi) = \text{sig}(\phi) \setminus \{d\}$$

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

PBES Manipulation

Removing redundancy from a PBES (step 2):

- Identify the **dependencies**:

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

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

PBES Manipulation

Removing redundancy from a PBES (step 3):

Assume the following PBES:

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

- **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 : $X_j(e) \in \text{dep}(\phi_j)$ 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:

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

- 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.

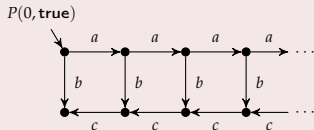
Note: typing of predicate variables in \mathcal{E} can differ from those in $\widehat{\mathcal{E}}$.

PBES Manipulation

Example

Consider the following process:

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



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

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

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

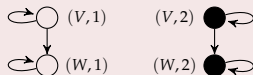
Instantiation of the PBES does not terminate.

PBES Manipulation

Example (Cont'd)

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

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



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:
$$\begin{aligned} (\nu V(d:\text{Bool}) &= (d \implies V(d)) \wedge W(d)) \\ (\mu W(d:\text{Bool}) &= d \vee (d \wedge W(d))) \end{aligned}$$
- Instantiation of the above PBES **terminates**

Outline

- 1 Transforming Satisfiability to Solving PBESs
- 2 Solving PBESs
- 3 PBES Manipulation
- 4 Exercise

Exercise

Consider the lossy channel system described by the following LPE:

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

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

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

Questions:

- Translate both formulae to PBESs given process $C(\text{true}, m_0)$
- 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 $M = Nat$ ($m_0 = 0$)