

Algorithms for Model Checking (2IMF35)

Lecture 10

Parameterised Boolean Equation Systems (2)

Background material:

Model Checking Processes with Data,

J.F. Groote and T.A.C. Willemse (*Sc. Comp. Progr. 2005*)

Proof Graphs for Parameterised Boolean Equation Systems,

S. Cranen, B. Luttik and T.A.C. Willemse (*CONCUR 2013*)

Tim Willemse

(t.a.c.willemse@tue.nl)

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

MF 6.073

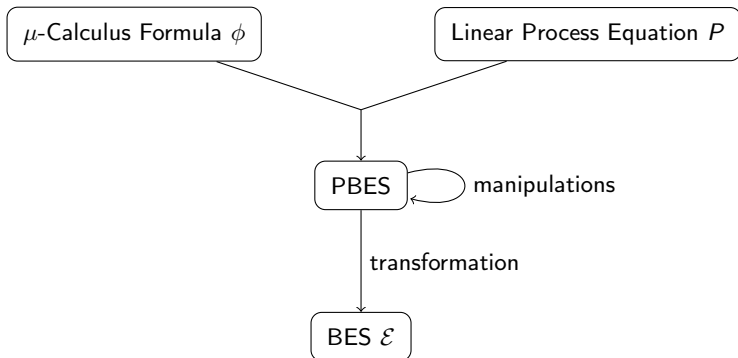
Parameterised Boolean Equation Systems

Verification via PBESs

Solving PBESs

Exercise

Verification Methodology:



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

Problem Description

1. Given a process $X(e)$ described by an LPE X over Act
2. Given a first-order modal μ -calculus formula ϕ
3. Given environments η, ε
4. Check whether $X(e) \models \phi$ holds, where:

$$X(e) \models \phi \text{ iff } e \in \llbracket \phi \rrbracket \eta \varepsilon$$

- ▶ Decidable for **finite data types**
 - Compute LTS $\llbracket X(e) \rrbracket$
 - Evaluate ϕ on $\llbracket X(e) \rrbracket$ using standard model checking algorithms
- ▶ In general **undecidable**
- ▶ Transform problem to **Parameterised** Boolean Equation Systems (PBESs)

Grammar for predicate formulae

$$\phi, \psi ::= b \mid X(e) \mid \phi \wedge \psi \mid \phi \vee \psi \mid \forall d : D. \phi \mid \exists d : D. \phi$$

- ▶ b is a **boolean expression** $n + m \geq 5$
- ▶ $X \in \mathcal{P}$ is a **sorted** predicate variable (or *relation*) $X:2^D$
- ▶ e is an expression of sort D
- ▶ Interpreting ϕ requires **two** environments ε (for data) and $\eta:\mathcal{P} \rightarrow 2^D$

Grammar for predicate formulae

$$\phi, \psi ::= b \mid X(e) \mid \phi \wedge \psi \mid \phi \vee \psi \mid \forall d : D. \phi \mid \exists d : D. \phi$$

- ▶ b is a **boolean expression** $n + m \geq 5$
- ▶ $X \in \mathcal{P}$ is a **sorted** predicate variable (or *relation*) $X : 2^D$
- ▶ e is an expression of sort D
- ▶ Interpreting ϕ requires **two** environments ε (for data) and $\eta : \mathcal{P} \rightarrow 2^D$

$$\llbracket b \rrbracket_{\eta \varepsilon} = \begin{cases} \text{true} & \text{if } \varepsilon(b) \\ \text{false} & \text{else} \end{cases}$$

Grammar for predicate formulae

$$\phi, \psi ::= b \mid X(e) \mid \phi \wedge \psi \mid \phi \vee \psi \mid \forall d : D. \phi \mid \exists d : D. \phi$$

- ▶ b is a **boolean expression** $n + m \geq 5$
- ▶ $X \in \mathcal{P}$ is a **sorted** predicate variable (or *relation*) $X:2^D$
- ▶ e is an expression of sort D
- ▶ Interpreting ϕ requires **two** environments ε (for data) and $\eta: \mathcal{P} \rightarrow 2^D$

$$\llbracket b \rrbracket_{\eta \varepsilon} = \begin{cases} \text{true} & \text{if } \varepsilon(b) \\ \text{false} & \text{else} \end{cases} \quad \llbracket X(e) \rrbracket_{\eta \varepsilon} = \begin{cases} \text{true} & \text{if } \varepsilon(e) \in \eta(X) \\ \text{false} & \text{else} \end{cases}$$

Grammar for predicate formulae

$$\phi, \psi ::= b \mid X(e) \mid \phi \wedge \psi \mid \phi \vee \psi \mid \forall d : D. \phi \mid \exists d : D. \phi$$

- ▶ b is a **boolean expression** $n + m \geq 5$
- ▶ $X \in \mathcal{P}$ is a **sorted** predicate variable (or *relation*) $X:2^D$
- ▶ e is an expression of sort D
- ▶ Interpreting ϕ requires **two** environments ε (for data) and $\eta: \mathcal{P} \rightarrow 2^D$

$$\llbracket b \rrbracket_{\eta \varepsilon} = \begin{cases} \text{true} & \text{if } \varepsilon(b) \\ \text{false} & \text{else} \end{cases} \quad \llbracket X(e) \rrbracket_{\eta \varepsilon} = \begin{cases} \text{true} & \text{if } \varepsilon(e) \in \eta(X) \\ \text{false} & \text{else} \end{cases}$$

$$\llbracket \phi \wedge \psi \rrbracket_{\eta \varepsilon} = \llbracket \phi \rrbracket_{\eta \varepsilon} \text{ and } \llbracket \psi \rrbracket_{\eta \varepsilon} \quad \llbracket \phi \vee \psi \rrbracket_{\eta \varepsilon} = \llbracket \phi \rrbracket_{\eta \varepsilon} \text{ or } \llbracket \psi \rrbracket_{\eta \varepsilon}$$

Grammar for predicate formulae

$$\phi, \psi ::= b \mid X(e) \mid \phi \wedge \psi \mid \phi \vee \psi \mid \forall d : D. \phi \mid \exists d : D. \phi$$

- ▶ b is a **boolean expression** $n + m \geq 5$
- ▶ $X \in \mathcal{P}$ is a **sorted** predicate variable (or *relation*) $X : 2^D$
- ▶ e is an expression of sort D
- ▶ Interpreting ϕ requires **two** environments ε (for data) and $\eta : \mathcal{P} \rightarrow 2^D$

$$\llbracket b \rrbracket_{\eta \varepsilon} = \begin{cases} \text{true} & \text{if } \varepsilon(b) \\ \text{false} & \text{else} \end{cases}$$

$$\llbracket X(e) \rrbracket_{\eta \varepsilon} = \begin{cases} \text{true} & \text{if } \varepsilon(e) \in \eta(X) \\ \text{false} & \text{else} \end{cases}$$

$$\llbracket \phi \wedge \psi \rrbracket_{\eta \varepsilon} = \llbracket \phi \rrbracket_{\eta \varepsilon} \text{ and } \llbracket \psi \rrbracket_{\eta \varepsilon}$$

$$\llbracket \phi \vee \psi \rrbracket_{\eta \varepsilon} = \llbracket \phi \rrbracket_{\eta \varepsilon} \text{ or } \llbracket \psi \rrbracket_{\eta \varepsilon}$$

$$\llbracket \forall d : D. \phi \rrbracket_{\eta \varepsilon} = \text{for all } v \in D: \\ \llbracket \phi \rrbracket_{\eta(\varepsilon[d := v])}$$

$$\llbracket \exists d : D. \phi \rrbracket_{\eta \varepsilon} = \text{for some } v \in D: \\ \llbracket \phi \rrbracket_{\eta(\varepsilon[d := v])}$$

A **parameterised Boolean equation** is an equation of the form $\sigma X(d : D) = \phi$

- ▶ σ is a least fixed point sign μ or a greatest fixed point sign ν .
- ▶ ϕ is a predicate formula, X a predicate variable
- ▶ a parameterised Boolean equation **system** is a sequence of such equations

- ▶ **bound (bnd), free, well-formedness, open, close, rank** as in BESs
- ▶ As in BESs, the **order** of equations is important.
- ▶ Assume, for simplicity, that all equations range over sort D only

The **solution** of a PBES \mathcal{E} is an environment: $\eta : \mathcal{P} \rightarrow 2^D$;

We define $\llbracket \mathcal{E} \rrbracket \eta \varepsilon$ by recursion on \mathcal{E} .

$$\left\{ \begin{array}{l} \llbracket \epsilon \rrbracket \eta \varepsilon \quad \quad \quad := \eta \\ \llbracket (\mu X(d : D) = \phi) \mathcal{E} \rrbracket \eta \varepsilon \quad := \llbracket \mathcal{E} \rrbracket \eta [X := \mu \Phi_{\mathcal{E}, \eta, \varepsilon}^{X, d}] \varepsilon \\ \llbracket (\nu X(d : D) = \phi) \mathcal{E} \rrbracket \eta \varepsilon \quad := \llbracket \mathcal{E} \rrbracket \eta [X := \nu \Phi_{\mathcal{E}, \eta, \varepsilon}^{X, d}] \varepsilon \end{array} \right.$$

Note: $\nu \Phi_{\mathcal{E}, \theta, \varepsilon}^{X, d}$ is the greatest fixpoint to the following monotone functional:

$$\Phi_{\mathcal{E}, \eta, \varepsilon}^{X, d}(Z) := \{v \in D \mid \llbracket \phi \rrbracket (\llbracket \mathcal{E} \rrbracket \eta [X := Z] \varepsilon) \varepsilon [d := v]\}$$

Let:

- ▶ $\eta : \mathcal{P} \rightarrow 2^D$ be a predicate environment and ε a data environment
- ▶ $\text{sig}(\mathcal{E}) = \{(X, v) \mid X \in \text{bnd}(\mathcal{E}), v \in D\}$ be the **signatures**

Definition (Signature Environments)

Assume $S \subseteq \text{sig}(\mathcal{E})$.

- ▶ S_{true} is the **environment** defined as $S_{\text{true}}(X) = \{v \mid (X, v) \in S\}$ for all X
- ▶ S_{false} is the **environment** defined as $S_{\text{false}}(X) = \{v \mid (X, v) \notin S\}$ for all X

Definition (Dependency Graphs)

Let b be a Boolean.

A structure $\langle S, R, L \rangle$ is a b -dependency graph for closed \mathcal{E} and data environment ε if:

- ▶ $S \subseteq \text{sig}(\mathcal{E})$
- ▶ $L(X, v) = \text{rank}(X)$
- ▶ $R \subseteq S \times S$ such that: **if** for $\sigma X(d : D) = \phi$ in \mathcal{E} , $(X, v) \in S$ **then**
 - If $b = \text{true}$, we require: $\llbracket \phi \rrbracket ((X, v)^\bullet)_{\text{true}\varepsilon}[d := v]$
 - If $b = \text{false}$, we require: $\neg \llbracket \phi \rrbracket ((X, v)^\bullet)_{\text{false}\varepsilon}[d := v]$

Where

- ▶ $(X, v)^\bullet = \{(Z, w) \in S \mid (X, v) R (Z, w)\}$

Example

Consider the following equation system \mathcal{E} .

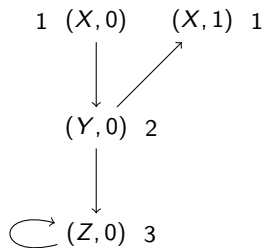
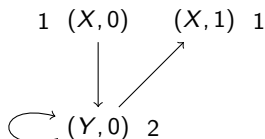
$$\mu X(b : \text{Bit}) = Y(b) \vee b = 1$$

$$\nu Y(b : \text{Bit}) = Y(b) \vee (X(1) \wedge Z(b))$$

$$\mu Z(b : \text{Bit}) = Z(b)$$

Below are two true-dependency graphs $\langle S, R, L \rangle$ with $(X, 0) \in S$.

Note that $\text{sig}(\mathcal{E}) = \{(U, 0), (U, 1) \mid U = X, Y, Z\}$.



Example (Continued)

To see that the graph on the left satisfies the true-dependency graph property:

$$\text{for } (X, 0): \llbracket Y(b) \vee b = 1 \rrbracket \{(Y, 0)\}_{\text{true}} \delta[b := 0] = \text{true}$$

$$\text{for } (X, 1): \llbracket Y(b) \vee b = 1 \rrbracket \emptyset_{\text{true}} \delta[b := 1] = \text{true}$$

$$\text{for } (Y, 0): \llbracket Y(b) \vee X(1) \rrbracket \{(Y, 0), (X, 1)\}_{\text{true}} \delta[b := 0] = \text{true}$$

- ▶ Any infinite path goes through states with label 2, hence, it satisfies the true-proof graph property.
- ▶ Note that in this true-dependency graph, $(Y, 0) \rightarrow (X, 1)$ can be left out, because the right hand side of the equation for Y is disjunctive and the left disjunct is true.

Definition (Proof Graphs)

A true-dependency graph $\langle S, R, L \rangle$ is a **proof graph** iff for all $s \in S$ and all **infinite** paths $\pi \in \text{path}(s)$:

$\min\{r \mid \text{label } r \text{ occurs infinitely often on } \pi\}$ is even

Theorem

For all closed PBESs \mathcal{E} and all η, ε :

$v \in \llbracket \mathcal{E} \rrbracket_{\eta \varepsilon}(X)$ iff there is a proof graph $\langle S, R, L \rangle$ such that $(X, v) \in S$

Dually:

Definition (Refutation Graphs)

A false-dependency graph $\langle S, R, L \rangle$ is a **refutation graph** iff for all $s \in S$ and all **infinite** paths $\pi \in \text{path}(s)$:

$\min\{r \mid \text{label } r \text{ occurs infinitely often on } \pi\}$ is odd

Theorem

For all closed PBESs \mathcal{E} and all η, ε :

$v \notin \llbracket \mathcal{E} \rrbracket_{\eta \varepsilon}(X)$ iff there is a refutation graph $\langle S, R, L \rangle$ such that $(X, v) \in S$

Example

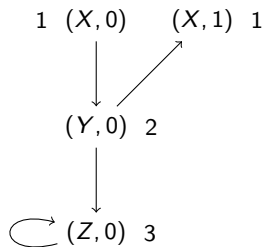
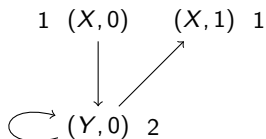
Consider the following equation system \mathcal{E} .

$$\mu X(b : \text{Bit}) = Y(b) \vee b = 1$$

$$\nu Y(b : \text{Bit}) = Y(b) \vee (X(1) \wedge Z(b))$$

$$\mu Z(b : \text{Bit}) = Z(b)$$

Below are two true-dependency graphs $\langle S, R, L \rangle$ with $(X, 0) \in S$. The left one is a true-proof graph; the right one is not.



Parameterised Boolean Equation Systems

Verification via PBESs

Solving PBESs

Exercise

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 \mathcal{E}
- ▶ 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

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

- ▶ The order of the equations respects the subterm ordering in $\sigma Z. \phi$
- ▶ Transformation is such that $X(e) \models \sigma Z. \phi$ iff $e \in \llbracket \mathcal{E} \rrbracket_{\eta \varepsilon}(\tilde{Z})$

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

Identify Bound Data Variables

$$\text{Par}(Z, b, I) = \text{Par}(Z, X, I) = \square$$

$$\text{Par}(Z, \phi \wedge \psi, I) = \text{Par}(Z, \phi \vee \psi, I) = \text{Par}(Z, \phi, I) ++ \text{Par}(Z, \psi, I)$$

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

$$\text{Par}(Z, [\alpha]\phi, I) = \text{Par}(Z, \langle \alpha \rangle \phi, I) = \text{Par}(Z, \phi, I)$$

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

Example

The one-place buffer system described by process B :

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

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

Create Equation System Outline

$$\mathbf{E}(b) = \epsilon$$

$$\mathbf{E}(Z) = \epsilon$$

$$\mathbf{E}(\phi \wedge \psi) = \mathbf{E}(\phi) \mathbf{E}(\psi)$$

$$\mathbf{E}(\phi \vee \psi) = \mathbf{E}(\phi) \mathbf{E}(\psi)$$

$$\mathbf{E}(\forall d':D'.\phi) = \mathbf{E}(\phi)$$

$$\mathbf{E}(\exists d':D'.\phi) = \mathbf{E}(\phi)$$

$$\mathbf{E}([\alpha]\phi) = \mathbf{E}(\phi)$$

$$\mathbf{E}(\langle \alpha \rangle \phi) = \mathbf{E}(\phi)$$

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

Example

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

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

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

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

RHS:

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

RHS:

$$\text{RHS}(b) = b$$

$$\text{RHS}(Z) = \check{Z}(d, \text{Par}(Z, \psi, []))$$

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

RHS:

$$\begin{array}{ll} \text{RHS}(b) & = b & \text{RHS}(Z) & = \check{Z}(d, \text{Par}(Z, \psi, [])) \\ \text{RHS}(\phi \wedge \psi) & = \text{RHS}(\phi) \wedge \text{RHS}(\psi) & \text{RHS}(\phi \vee \psi) & = \text{RHS}(\phi) \vee \text{RHS}(\psi) \end{array}$$

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

RHS:

| | | | |
|-----------------------------------|--|-----------------------------------|--|
| $\text{RHS}(b)$ | $= b$ | $\text{RHS}(Z)$ | $= \check{Z}(d, \text{Par}(Z, \psi, []))$ |
| $\text{RHS}(\phi \wedge \psi)$ | $= \text{RHS}(\phi) \wedge \text{RHS}(\psi)$ | $\text{RHS}(\phi \vee \psi)$ | $= \text{RHS}(\phi) \vee \text{RHS}(\psi)$ |
| $\text{RHS}(\forall d':D'. \phi)$ | $= \forall d':D'. \text{RHS}(\phi)$ | $\text{RHS}(\exists d':D'. \phi)$ | $= \exists d':D'. \text{RHS}(\phi)$ |

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

RHS:

| | | | |
|----------------------------------|--|----------------------------------|--|
| $\text{RHS}(b)$ | $= b$ | $\text{RHS}(Z)$ | $= \tilde{Z}(d, \text{Par}(Z, \psi, []))$ |
| $\text{RHS}(\phi \wedge \psi)$ | $= \text{RHS}(\phi) \wedge \text{RHS}(\psi)$ | $\text{RHS}(\phi \vee \psi)$ | $= \text{RHS}(\phi) \vee \text{RHS}(\psi)$ |
| $\text{RHS}(\forall d':D'.\phi)$ | $= \forall d':D'. \text{RHS}(\phi)$ | $\text{RHS}(\exists d':D'.\phi)$ | $= \exists d':D'. \text{RHS}(\phi)$ |
| $\text{RHS}(\sigma Z.\phi)$ | $= \tilde{Z}(d, \text{Par}(Z, \psi, []))$ | | |

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

RHS:

$$\text{RHS}(b) = b \qquad \text{RHS}(Z) = \tilde{Z}(d, \text{Par}(Z, \psi, []))$$

$$\text{RHS}(\phi \wedge \psi) = \text{RHS}(\phi) \wedge \text{RHS}(\psi) \qquad \text{RHS}(\phi \vee \psi) = \text{RHS}(\phi) \vee \text{RHS}(\psi)$$

$$\text{RHS}(\forall d':D'. \phi) = \forall d':D'. \text{RHS}(\phi) \qquad \text{RHS}(\exists d':D'. \phi) = \exists d':D'. \text{RHS}(\phi)$$

$$\text{RHS}(\sigma Z. \phi) = \tilde{Z}(d, \text{Par}(Z, \psi, []))$$

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

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

RHS:

$$\text{RHS}(b) = b \qquad \text{RHS}(Z) = \tilde{Z}(d, \text{Par}(Z, \psi, []))$$

$$\text{RHS}(\phi \wedge \psi) = \text{RHS}(\phi) \wedge \text{RHS}(\psi) \qquad \text{RHS}(\phi \vee \psi) = \text{RHS}(\phi) \vee \text{RHS}(\psi)$$

$$\text{RHS}(\forall d':D'. \phi) = \forall d':D'. \text{RHS}(\phi) \qquad \text{RHS}(\exists d':D'. \phi) = \exists d':D'. \text{RHS}(\phi)$$

$$\text{RHS}(\sigma Z. \phi) = \tilde{Z}(d, \text{Par}(Z, \psi, []))$$

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

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

Example (Verification of the Buffer process B , continued)

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

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

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

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

Matching parameterised actions with action formulae:

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

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

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

Property 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):

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

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

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

Parameterised Boolean Equation Systems

Verification via PBESs

Solving PBESs

Exercise

How to solve PBESs

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

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

- ▶ Gauß Elimination on PBES + symbolic approximation of equations
- ▶ Instantiation to BES and subsequently solve the BES
- ▶ 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

$$\forall \varepsilon, \eta : \llbracket \phi \rrbracket \eta \varepsilon = \llbracket \psi \rrbracket \eta \varepsilon$$

- ▶ 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 PBES manipulations/solving techniques.

Gauß elimination on PBESs + Symbolic Approximation:

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

- **Local solution:** eliminate X in its defining equation:

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

- X^ω can be found by **symbolic approximation**:
- $X^0 = \text{false}$ if $\sigma = \mu$, else $X^0 = \text{true}$
- $X^{n+1} = \phi[X := X^n]$
- X^ω may require **transfinite approximation**; else $X^\omega = X^n$ for $X^n \leftrightarrow X^{n+1}$

Gauß elimination on PBESs + Symbolic Approximation:

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

- **Local solution:** eliminate X in its defining equation:

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

- X^ω can be found by **symbolic approximation**:
 - $X^0 = \text{false}$ if $\sigma = \mu$, else $X^0 = \text{true}$
 - $X^{n+1} = \phi[X := X^n]$
 - X^ω may require **transfinite approximation**; else $X^\omega = X^n$ for $X^n \leftrightarrow X^{n+1}$
- Substitute **definition backwards**:
- $\mathcal{E}_0 (\sigma_1 X_1(d_1 : D_1) = \phi_1) \mathcal{E}_1 (\sigma_2 X_2(d_2 : D_2) = \phi_2) \mathcal{E}_2$
 becomes: $\mathcal{E}_0 (\sigma_1 X_1(d_1 : D_1) = \phi_1 [X_2 := \phi_2]) \mathcal{E}_1 (\sigma_2 X_2(d_2 : D_2) = \phi_2) \mathcal{E}_2$

Gauß elimination on PBESs + Symbolic Approximation:

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

- ▶ **Local solution:** eliminate X in its defining equation:

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

- X^ω can be found by **symbolic approximation**:
 - $X^0 = \text{false}$ if $\sigma = \mu$, else $X^0 = \text{true}$
 - $X^{n+1} = \phi[X := X^n]$
 - X^ω may require **transfinite approximation**; else $X^\omega = X^n$ for $X^n \leftrightarrow X^{n+1}$
- ▶ Substitute **definition backwards**:

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

- ▶ Substitute **solved** equations (i.e. **not containing predicate variables**) **forward**:

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

Example

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

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

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

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

$$X^0 \equiv \text{true}$$

$$X^1 \equiv n \leq 2 \wedge (\text{odd}(n) \vee \text{true}) \leftrightarrow n \leq 2$$

$$X^2 \equiv n \leq 2 \wedge (\text{odd}(n) \vee n+1 \leq 2) \leftrightarrow n \leq 2 \wedge (\text{odd}(n) \vee n \leq 1) \leftrightarrow n \leq 1$$

$$X^3 \equiv n \leq 2 \wedge (\text{odd}(n) \vee n+1 \leq 1) \leftrightarrow n \leq 2 \wedge (\text{odd}(n) \vee n = 0) \leftrightarrow n \leq 1$$

So, solution to X is $n \leq 1$ (i.e., X semantically consists of the set $\{0, 1\}$)

Gauß Elimination terminates; **symbolic approximation** may not terminate

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

Parameterised Boolean Equation Systems

Verification via PBESs

Solving PBESs

Exercise

Consider the lossy channel system described by the following LPE:

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

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

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

Questions:

- ▶ Explain the first formula in natural language
- ▶ Translate both formulae to PBESs given process C
- ▶ Use Gauß Elimination to solve the PBES
- ▶ For which initial states of C do the properties hold?