

Algorithms for Model Checking (2IW55)

Lecture 14

Parameterised Boolean Equation Systems

Tim Willemse

(timw@win.tue.nl)

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

HG 6.81

Outline

Symbolic System Specification

First-order Modal μ -Calculus

Parameterised Boolean Equation Systems

Transforming Satisfiability to Solving PBESs

Exercise

Symbolic System Specification

System specification represented by **first-order formulae**:

- ▶ V, V' are sets of **variables** v_0, v'_0, v_1, v'_1
- ▶ D is the **domain** of these variables \mathbb{N}, \mathbb{R}
- ▶ **states** set of all mappings $V \rightarrow D$: $[V \rightarrow D]$
- ▶ **transitions** formulae $\mathcal{R}(V, V')$

Symbolic System Specification

Today: System specification represented by **Linear Process Equations**:

- ▶ **abstract data types** for reasoning about data
 - **data sorts** $Bool, Nat$
 - **function symbols** $and : Bool \times Bool \rightarrow Bool$
 - **equations** $and(x, true) = x$
- ▶ **process algebra** for reasoning about dynamic behaviour
 - **parameterised atomic actions** $read(n), write(n)$
 - **process operators** $+, \sum_{n:Nat}, \cdot$
 - **parameterised recursion** $X(n : Nat) = a \cdot X(n) + \sum_{m:Nat} b(m) \cdot X(m)$

Symbolic System Specification

Linear Process Equation format

$$\begin{aligned} X(d : D) = & \sum_{e_1 : D_1} c_1(d, e_1) \longrightarrow a_1(f_1(d, e_1)) \cdot X(g_1(d, e_1)) \\ & + \dots \\ & + \sum_{e_n : D_n} c_n(d, e_n) \longrightarrow a_n(f_n(d, e_n)) \cdot X(g_n(d, e_n)) \end{aligned}$$

- ▶ d is a vector of **state variables**
- ▶ e_i is the vector of **local variables** for summand i
- ▶ c_i is the **enabling condition** for summand i ; free variables in c_i are d and e_i
- ▶ $a_i \in Act$ is the (visible/invisible) **action label** for summand i
- ▶ f_i is the **parameter** for action a_i ; free variables in f_i are d and e_i
- ▶ g_i is the **next-state** function for summand i ; free variables in c_i are d and e_i

Symbolic System Specification

Linear Process Equation format

$$\begin{aligned}
 X(d : D) = & \sum_{e_1 : D_1} c_1(d, e_1) \longrightarrow a_1(f_1(d, e_1)) \cdot X(g_1(d, e_1)) \\
 & + \dots \\
 & + \sum_{e_n : D_n} c_n(d, e_n) \longrightarrow a_n(f_n(d, e_n)) \cdot X(g_n(d, e_n))
 \end{aligned}$$

Semantics: $[X(e)]$ defines the **Labelled Transition System** $[X(e)] = \langle S, s_0, Act', \rightarrow \rangle$ where:

- ▶ $S = \{X(d) \mid d \in D\}$ is the **state space**
- ▶ $s_0 = X(e)$ is the **initial state**
- ▶ $Act' = \{a_i(d) \mid 1 \leq i \leq n \wedge d \in D_{a_i}\}$ is the **set of actions**
- ▶ $X(d) \xrightarrow{a} X(d')$ **iff for some i : $\exists e_i : D_i. c_i(d, e_i) \wedge d' = g_i(d, e_i) \wedge a = a_i(f_i(d), e_i)$**

Symbolic System Specification

Example

Consider the system $X(0, \text{true})$ given by the following LPE:

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

Intuition:

- if b holds, then an **arbitrary** natural number can be read through action r
- if $\neg b$ holds, then **value n** is sent through action s

Formally:

- States: $X(n, b)$ for $(n, b) \in \text{Nat} \times \text{Bool}$
- Transitions: for all $n, m \in \text{Nat}$:

$$X(n, \text{true}) \xrightarrow{r(m)} X(m, \text{false}) \text{ and } X(n, \text{false}) \xrightarrow{s(n)} X(n, \text{true})$$

Outline

Symbolic System Specification

First-order Modal μ -Calculus

Parameterised Boolean Equation Systems

Transforming Satisfiability to Solving PBESs

Exercise

First-order Modal μ -Calculus

- ▶ Propositional Modal μ -calculus **subsumes** LTL, CTL, CTL*
- ▶ Propositions and action labels are **first class objects**

Problem

How to verify that any natural number n that is **read** through action r is **immediately sent** through action s in $X(0, \text{true})$:

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

- ▶ μ -Calculus formulae are **finite**
- ▶ $\nu X.[r(0)]\langle s(0) \rangle X \wedge [r(1)]\langle s(1) \rangle X \wedge \dots$ is **not** a μ -Calculus formula

First-order Modal mu-Calculus

First-order Modal mu-Calculus grammar

State formulae directly in **Positive Normal Form**:

$$\phi ::= \text{true} \mid \text{false} \mid b \mid Z(e) \mid \phi \vee \phi \mid \phi \wedge \phi \mid \exists d : D. \phi \mid \forall d : D. \phi \mid [\alpha]\phi \mid \langle \alpha \rangle \phi \mid (\mu Z(d : D := e). \phi) \mid (\nu Z(d : D := e). \phi)$$

Action formulae:

$$\alpha ::= \text{true} \mid b \mid a(e) \mid \neg \alpha \mid \alpha \wedge \alpha \mid \forall d : D. \alpha$$

- ▶ $a \in Act$ is an **action label**
- ▶ d is a vector of **bound** variables
- ▶ Z is a **parameterised predicate variable**
- ▶ b is a **boolean expression** $d \geq 3, \text{odd}(d)$
- ▶ e is an **expression** $3 + 5, d \geq 3, d + e$

First-order Modal mu-Calculus

Let $X(d : D) = \sum_{i \leq n} \sum_{e_i : D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$ be an LPE over *Act*

- ▶ Action formulae encode **possibly infinite sets of actions**
- ▶ Action formulae can contain **free data variables**
- ▶ $\exists n : Nat. r(n)$ encodes the set $\{r(n) \mid n \in Nat\}$
- ▶ $\langle \exists n : Nat. (\text{odd}(n) \wedge r(n)) \rangle \phi$: an **odd natural** can be read through action r

Action formulae are interpreted in the context of a **data environment** ε :

$$\begin{aligned}
 [\text{true}] \varepsilon &= \{a_i(d) \mid 1 \leq i \leq n \wedge d \in D_{a_i}\} \\
 [a(e)] \varepsilon &= \{a(d) \mid d = [e] \varepsilon\} \\
 [\neg \alpha] \varepsilon &= [\text{true}] \varepsilon \setminus [\alpha] \varepsilon \\
 [\alpha_1 \wedge \alpha_2] \varepsilon &= [\alpha_1] \varepsilon \cap [\alpha_2] \varepsilon \\
 [\forall d : D. \alpha] \varepsilon &= \bigcap_{v \in D} [\alpha] \varepsilon [d := v]
 \end{aligned}$$

For $[b] \varepsilon$ and $[e] \varepsilon$: replace each occurrence of a variable d with $\varepsilon(d)$

First-order Modal mu-Calculus

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

- ▶ **State formulae** are interpreted in the context of **two** environments
- ▶ A predicate environment θ assigns a **function** $D \rightarrow 2^{\{X(d) \mid d \in D\}}$ to every predicate variable

$$[\text{true}]_{\theta \varepsilon} = \{X(d) \mid d \in D\}$$

$$[\text{false}]_{\theta \varepsilon} = \emptyset$$

$$[b]_{\theta \varepsilon} = \begin{cases} [\text{true}]_{\theta \varepsilon} & \text{if } [b]_{\varepsilon} \text{ holds,} \\ [\text{false}]_{\theta \varepsilon} & \text{otherwise} \end{cases}$$

$$[Z(e)]_{\theta \varepsilon} = \theta(Z)([e]_{\varepsilon})$$

$$[\phi_1 \vee \phi_2]_{\theta \varepsilon} = [\phi_1]_{\theta \varepsilon} \cup [\phi_2]_{\theta \varepsilon}$$

$$[\phi_1 \wedge \phi_2]_{\theta \varepsilon} = [\phi_1]_{\theta \varepsilon} \cap [\phi_2]_{\theta \varepsilon}$$

First-order Modal mu-Calculus

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

- ▶ **State formulae** are interpreted in the context of **two** environments
- ▶ A predicate environment θ assigns a **function** $D \rightarrow 2^{\{X(d) \mid d \in D\}}$ to every predicate variable

$$[\forall d : D. \phi] \theta \varepsilon = \bigcap_{v \in D} [\phi] \theta \varepsilon [d := v] \dots \dots \dots \text{variable } d \text{ gets value } v$$

$$[\exists d : D. \phi] \theta \varepsilon = \bigcup_{v \in D} [\phi] \theta \varepsilon [d := v] \dots \dots \dots \text{variable } d \text{ gets value } v$$

$$[\langle \alpha \rangle \phi] \theta \varepsilon = \{X(d) \mid \exists d' \in D. \exists a \in \text{Act}. \exists v_a \in D_a. \\ X(d) \xrightarrow{a(v_a)} X(d') \wedge a(v_a) \in [\alpha] \varepsilon \wedge X(d') \in [\phi] \theta \varepsilon\}$$

$$[[\alpha] \phi] \theta \varepsilon = \{X(d) \mid \forall d' \in D. \forall a \in \text{Act}. \forall v_a \in D_a. \\ (X(d) \xrightarrow{a(v_a)} X(d') \wedge a(v_a) \in [\alpha] \varepsilon) \Rightarrow X(d') \in [\phi] \theta \varepsilon\}$$

First-order Modal mu-Calculus

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

- ▶ **State formulae** are interpreted in the context of **two** environments
 - ▶ A predicate environment θ assigns a **function** $D \rightarrow 2^{\{X(d) \mid d \in D\}}$ to every predicate variable
 - The set $([D \rightarrow 2^{\{X(d) \mid d \in D\}}], \sqsubseteq)$ is a **complete lattice**:
 - For $f, g : D \rightarrow 2^{\{X(d) \mid d \in D\}}$, $f \sqsubseteq g$ iff $\forall v \in D : f(v) \subseteq g(v)$
 - Let $\Phi_{\theta \varepsilon} := \lambda f : D \rightarrow 2^{\{X(d) \mid d \in D\}}. (\lambda v \in D. [\phi] (\theta[Z := f])(\varepsilon[d := v]))$
 - $\Phi_{\theta \varepsilon}$ is **monotone**: $f \sqsubseteq g$ implies $\Phi_{\theta \varepsilon}(f) \sqsubseteq \Phi_{\theta \varepsilon}(g)$
- \Rightarrow least and greatest fixpoints in $([D \rightarrow 2^{\{X(d) \mid d \in D\}}], \sqsubseteq)$ **exist**

$$[(\nu Z(d : D := e). \phi)]\theta \varepsilon = \mathbf{GFP}(\Phi_{\theta \varepsilon})([e]\varepsilon)$$

$$[(\mu Z(d : D := e). \phi)]\theta \varepsilon = \mathbf{LFP}(\Phi_{\theta \varepsilon})([e]\varepsilon)$$

First-order Modal mu-Calculus

Example

1. Absence of deadlock:

$\nu X. [\text{true}]X \wedge \langle \text{true} \rangle \text{true}$ always **some** action enabled

2. Reading value n can always immediately be followed by sending n :

$\nu X. [\neg(\exists m : \text{Nat}. r(m))]X \wedge \forall n : \text{Nat}. [r(n)]\langle s(n) \rangle \text{true}$

3. The values consecutively received over action r are **ascending**:

$\nu X(n : \text{Nat} := 0). [\neg(\exists m : \text{Nat}. r(m))]X(n) \wedge \forall m : \text{Nat}. [r(m)](m \geq n \wedge X(m))$

- any value received is always **at least 0** therefore $\nu X(n : \text{Nat} := 0)$
- the set of read actions $\exists m : \text{Nat}. r(m)$
- non-read actions are "ignored" $[\neg(\exists m : \text{Nat}. r(m))]X(n)$
- a state satisfying $X(m)$ can only receive values **at least m**
- read action $r(m)$ must lead to a state satisfying **$m \geq n$ and $X(m)$**

Outline

Symbolic System Specification

First-order Modal μ -Calculus

Parameterised Boolean Equation Systems

Transforming Satisfiability to Solving PBESs

Exercise

Parameterised Boolean Equation Systems

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 θ, ε
4. Check whether $X(e) \models \phi$ holds, where:

$$X(e) \models \phi \text{ iff } X(e) \in [\phi]\theta\varepsilon$$

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

Parameterised Boolean Equation Systems

Grammar for PBESs: $\mathcal{E} ::= (\mu X(d : D) = \phi) \mathcal{E} \mid (\nu X(d : D) = \phi) \mathcal{E} \mid \epsilon$

- ▶ X is a **parameterised predicate variable**
- ▶ d is a vector of **data variables** of sort D
- ▶ ϕ is a **predicate formula**
- ▶ ϵ is the empty equation system (usually omitted)

Grammar for predicate formulae (directly in **Positive Normal Form**)

$$\phi ::= \text{true} \mid \text{false} \mid b \mid X(e) \mid \phi \wedge \phi \mid \phi \vee \phi \mid \forall d : D. \phi \mid \exists d : D. \phi$$

- ▶ b is a **boolean expression**

Example

$$(\nu X(n : \text{Nat}) = n \geq 5 \wedge Y(n, \text{true})) (\mu Y(m : \text{Nat}, b : \text{Bool}) = b \vee m \leq 10 \vee X(m + 1))$$

Parameterised Boolean Equation Systems

Semantics (1)

- ▶ Predicate formulae can contain data variables and predicate variables
- ▶ Interpretation again w.r.t. a data environment ε and a predicate environment η
- ▶ Predicate environment maps a variable X to a **function from the set of functions** $[D \rightarrow Bool]$

$$[true]_{\eta\varepsilon} = true$$

$$[false]_{\eta\varepsilon} = false$$

$$[b]_{\eta\varepsilon} = true \text{ if } [b]_{\varepsilon} \text{ holds, else false}$$

$$[X(e)]_{\eta\varepsilon} = \eta(X)([e]_{\varepsilon})$$

$$[\phi_1 \wedge \phi_2]_{\eta\varepsilon} = [\phi_1] \text{ and } [\phi_2]$$

$$[\phi_1 \vee \phi_2]_{\eta\varepsilon} = [\phi_1] \text{ or } [\phi_2]$$

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

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

Parameterised Boolean Equation Systems

Semantics (2)

- ▶ As in BESs, the **order** of equations is important.
- ▶ **bounded, free, well-formedness, open, close** as in BESs
- ▶ The **solution** of a PBES is an environment: $\eta : Var \rightarrow (D \rightarrow Bool)$

Given a PBES \mathcal{E} , we define $[\mathcal{E}] : Val \rightarrow Val$ by recursion on \mathcal{E} .

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

- ▶ $[\phi]\eta_\mu\epsilon := \text{LFP}(\lambda f : D \rightarrow Bool. \lambda v \in D. [\phi]\eta[X := f]\epsilon[d := v])$
- ▶ $[\phi]\eta_\nu\epsilon := \text{GFP}(\lambda f : D \rightarrow Bool. \lambda v \in D. [\phi]\eta[X := f]\epsilon[d := v])$

Outline

Symbolic System Specification

First-order Modal μ -Calculus

Parameterised Boolean Equation Systems

Transforming Satisfiability to Solving PBESs

Exercise

Transforming Satisfiability to Solving PBESs

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

- ▶ 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 :

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

- ▶ We define a PBES \mathcal{E} with the following property:

$$([\mathcal{E}]_{\eta\epsilon})(X)(e, e_f) = \text{true iff } X(e) \models \sigma Z(d_f : D_f := e_f). \phi$$

- ▶ The transformation is similar to the transformation to BES:
 - For each subformula $\sigma Z(d_f : D_f := e_f). \phi$, we add an equation

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

- $Par(Z, \psi)$ contains the smallest **vector of variables (+ their sorts)** that **may occur free** within the scope of $\sigma Z \dots$ in the **original formula ψ**
- The order of the equations respects the subterm ordering in ψ

Transforming Satisfiability to Solving PBESs

- ▶ 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 :

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

- ▶ Operator $\mathbf{E}(\psi)$ breaks down the structure of ψ and **generates equations**

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

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

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

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

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

Transforming Satisfiability to Solving PBESs

- ▶ 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 :

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

- ▶ Operator **RHS**(ψ) breaks down the structure of ψ and **generates predicates**

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

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

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

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

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

Transforming Satisfiability to Solving PBESs

- ▶ 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 :

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

- ▶ Operator $\mathbf{RHS}(\psi)$ breaks down the structure of ψ and **generates predicates**

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

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

Transforming Satisfiability to Solving PBESs

Matching parameterised actions with action formulae can also be transformed to a predicate

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

- ▶ $\text{match}(a_i(d_{a_i}))$ 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
- ▶ $\text{match}(a_i(d_{a_i}))$ **will not cause problems** on the left-hand side of an implication

Example

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

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

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

Resulting PBES: introduce an auxiliary fixed point A:

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

Follow PBES translation rules:

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

Outline

Symbolic System Specification

First-order Modal μ -Calculus

Parameterised Boolean Equation Systems

Transforming Satisfiability to Solving PBESs

Exercise

Exercise

Given an arbitrary formula α and a state formula ϕ in which variable d does not occur. Are the following pairs of formulae equivalent? If not, give a model in which one holds and the other does not. Also explain whether one formula is stronger than the other or whether they are incomparable.

- ▶ $\langle \exists d:D.\alpha \rangle \phi$ and $\exists d:D.\langle \alpha \rangle \phi$
- ▶ $[\exists d:D.\alpha] \phi$ and $\forall d:D.[\alpha] \phi$
- ▶ $\exists d:D.[\alpha] \phi$ and $[\forall d:D.\alpha] \phi$
- ▶ $\langle \forall d:D.\alpha \rangle \phi$ and $\forall d:D.\langle \alpha \rangle \phi$