

## Algorithms for Model Checking (2IW55)

### Lecture 13

Parameterised Boolean Equation Systems

Background material:

"Model-checking processes with data" and  
"Parameterised Boolean Equation Systems",  
J.F. Groote and T.A.C. Willemse

Tim Willemse

(timw@win.tue.nl)

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

HG 6.81

## Outline

- 1 Symbolic System Specification
- 2 First-order Modal  $\mu$ -Calculus
- 3 Parameterised Boolean Equation Systems
- 4 Transforming Satisfiability to Solving PBESs
- 5 Exercise

# 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**  $[X(e)] = \langle S, s_0, \text{Act}', \rightarrow \rangle$  where:

- $S = D$  is the **state space**
- $s_0 = e$  is the **initial state**
- $\text{Act}' = \{a_i(d) \mid 1 \leq i \leq n \wedge d \in D_{a_i}\}$  is the **set of actions**
- $d \xrightarrow{a} d'$  **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:

- State space:  $\text{Nat} \times \text{Bool}$
- Transitions: for all  $n, m \in \text{Nat}$ :  
 $(n, \text{true}) \xrightarrow{r(m)} (m, \text{false})$  and  $(n, \text{false}) \xrightarrow{s(n)} (n, \text{true})$

## Outline

- 1 Symbolic System Specification
- 2 **First-order Modal  $\mu$ -Calculus**
- 3 Parameterised Boolean Equation Systems
- 4 Transforming Satisfiability to Solving PBESs
- 5 Exercise

First-order Modal  $\mu$ -Calculus

- Propositional Modal  $\mu$ -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  $X(0, \text{true})$ :

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

- $\mu$ -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  $\mu$ -Calculus formula



## First-order Modal mu-Calculus

## First-order Modal mu-Calculus grammar

State formulae directly in Positive Normal Form:

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

Action formulae:

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

- $a \in \text{Act}$  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**  $\varepsilon$ :

$$\begin{aligned}
 [b]\varepsilon &= \begin{cases} \emptyset & \text{if not } \varepsilon(b) \\ \{a_i(d) \mid 1 \leq i \leq n \wedge d \in D_{a_i}\} & \text{else} \end{cases} \\
 [a(e)]\varepsilon &= \{a(d) \mid d = \varepsilon(e)\varepsilon\} \\
 [\neg\alpha]\varepsilon &= [\text{true}]\varepsilon \setminus [\alpha]\varepsilon \\
 [\alpha_1 \wedge \alpha_2]\varepsilon &= [\alpha_1]\varepsilon \cap [\alpha_2]\varepsilon \\
 [\forall d : D. \alpha]\varepsilon &= \bigcap_{v \in D} [\alpha]\varepsilon[d := v]
 \end{aligned}$$

## 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  $\theta$  assigns a **function**  $D \rightarrow 2^D$  to every predicate variable

$$[b]\theta\varepsilon = \begin{cases} D & \text{if } \varepsilon(b) \text{ holds,} \\ \emptyset & \text{otherwise} \end{cases}$$

$$[Z(e)]\theta\varepsilon = \theta(Z)(\varepsilon(e))$$

$$[\phi_1 \vee \phi_2]\theta\varepsilon = [\phi_1]\theta\varepsilon \cup [\phi_2]\theta\varepsilon$$

$$[\phi_1 \wedge \phi_2]\theta\varepsilon = [\phi_1]\theta\varepsilon \cap [\phi_2]\theta\varepsilon$$

## 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  $\mathbf{Act}$

- **State formulae** are interpreted in the context of **two** environments
- A predicate environment  $\theta$  assigns a **function**  $D \rightarrow 2^D$  to every predicate variable

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

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

$$[\langle \alpha \rangle \phi] \theta \varepsilon = \{d \mid \exists d' \in D. \exists a \in \mathbf{Act}. \exists v_a \in D_a.$$

$$d \xrightarrow{a(v_a)} d' \wedge a(v_a) \in [\alpha] \varepsilon \wedge d' \in [\phi] \theta \varepsilon\}$$

$$[[\alpha] \phi] \theta \varepsilon = \{d \mid \forall d' \in D. \forall a \in \mathbf{Act}. \forall v_a \in D_a.$$

$$(d \xrightarrow{a(v_a)} d' \wedge a(v_a) \in [\alpha] \varepsilon) \Rightarrow d' \in [\phi] \theta \varepsilon\}$$

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  $\theta$  assigns a **function**  $D \rightarrow 2^D$  to every predicate variable

- The set  $([D \rightarrow 2^D], \sqsubseteq)$  is a **complete lattice**:
- For  $f, g : D \rightarrow 2^D$ ,  $f \sqsubseteq g$  iff  $\forall v \in D : f(v) \subseteq g(v)$
- Let  $\Phi_{\theta_\varepsilon} := \lambda f : D \rightarrow 2^D. (\lambda v \in D. [\phi](\theta[Z := f])(\varepsilon[d := v]))$
- $\Phi_{\theta_\varepsilon}$  is **monotone**:  $f \sqsubseteq g$  implies  $\Phi_{\theta_\varepsilon}(f) \sqsubseteq \Phi_{\theta_\varepsilon}(g)$

It follows that least and greatest fixpoints in  $([D \rightarrow 2^D], \sqsubseteq)$  **exist**

$$\begin{aligned} [(\nu Z(d:D := e). \phi)]\theta_\varepsilon &= \mathbf{gfp}(\Phi_{\theta_\varepsilon})([e]\varepsilon) \\ [(\mu Z(d:D := e). \phi)]\theta_\varepsilon &= \mathbf{lfp}(\Phi_{\theta_\varepsilon})([e]\varepsilon) \end{aligned}$$

## 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

- 1 Symbolic System Specification
- 2 First-order Modal  $\mu$ -Calculus
- 3 Parameterised Boolean Equation Systems**
- 4 Transforming Satisfiability to Solving PBESs
- 5 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  $\mu$ -calculus formula  $\phi$
- 3 Given environments  $\theta, \varepsilon$
- 4 Check whether  $X(e) \models \phi$  holds, where:

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

- Decidable for **finite data types**
  - Compute LTS  $[X(e)]$
  - Evaluate  $\phi$  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  $D$
- $\phi$  is a **predicate formula**
- $\epsilon$  is the empty equation system (usually omitted)

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

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

- $b$  is a **boolean expression**

### Example

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

## Parameterised Boolean Equation Systems

## Semantics (1)

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

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

$$[X(e)]\eta\varepsilon \quad = \eta(X)(\varepsilon(e))$$

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

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

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

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

## 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  $\mathcal{E}$ .

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

- $[\phi]\eta_\mu\mathcal{E} := \text{lfp}(\lambda f : D \rightarrow Bool. \lambda v \in D. [\phi]\eta[X := f]\mathcal{E}[d := v])$
- $[\phi]\eta_\nu\mathcal{E} := \text{gfp}(\lambda f : D \rightarrow Bool. \lambda v \in D. [\phi]\eta[X := f]\mathcal{E}[d := v])$

## Outline

- 1 Symbolic System Specification
- 2 First-order Modal  $\mu$ -Calculus
- 3 Parameterised Boolean Equation Systems
- 4 Transforming Satisfiability to Solving PBESs**
- 5 Exercise

## Transforming Satisfiability to Solving PBESs

### Transformation of the First-order Modal $\mu$ -Calculus to PBES

- Given is a First-order Modal  $\mu$ -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:

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

- 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  $\psi$**
- The order of the equations respects the subterm ordering in  $\psi$

## Transforming Satisfiability to Solving PBESs

- Given is a First-order Modal  $\mu$ -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  $\psi$  and **generates equations**

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

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

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

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

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

## Transforming Satisfiability to Solving PBESs

- Given is a First-order Modal  $\mu$ -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  $\psi$  and **generates predicates**

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

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

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

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

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

## Transforming Satisfiability to Solving PBESs

- Given is a First-order Modal  $\mu$ -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  $\psi$  and **generates predicates**

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

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



## 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

## Transforming Satisfiability to Solving PBESs

## 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

- 1 Symbolic System Specification
- 2 First-order Modal  $\mu$ -Calculus
- 3 Parameterised Boolean Equation Systems
- 4 Transforming Satisfiability to Solving PBESs
- 5 Exercise

## Exercise

Given an arbitrary formula  $\alpha$  and a state formula  $\phi$  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  $\forall d:D.\langle \alpha \rangle \phi$