Algorithms for Model Checking (2IW55)

Lecture 9

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

Symbolic System Specification

Today: System specification represented by Linear Process Equations:

- abstract data types for reasoning about data
  - data sorts ..... $Bool, Nat$
  - function symbols ..... $and : Bool \times Bool \to Bool$
  - equations ..... $and(x, \mathsf{true}) = x$
- 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

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

- 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
- $a_i \in \mathit{Act}$ 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

Example

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

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

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

Symbolic System Specification

Linear Process Equation format

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

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

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

Outline

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

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

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

First-order Modal mu-Calculus

First-order Modal mu-Calculus grammar

State formulae directly in Positive Normal Form:

$$\begin{aligned}\phi ::= \quad & b \mid Z(e) \mid \phi \vee \phi \mid \phi \wedge \phi \mid \exists d{:}D.\ \phi \mid \forall d{:}D.\ \phi \mid \\ & [\alpha]\phi \mid \langle\alpha\rangle\phi \mid (\mu Z(d{:}D := e).\ \phi) \mid (\nu Z(d{:}D := e).\ \phi)\end{aligned}$$

Action formulae:

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

- $a \in \mathit{Act}$ is an action label
- d is a vector of bound variables
- Z is a parameterised predicate variable
- b is a boolean expression ..... $d \geq 3, \mathsf{odd}(d)$
- 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 $\{r(n) \mid n \in Nat\}$
- $\langle \exists n{:}Nat.\ (\mathsf{odd}(n) \wedge r(n)) \rangle \phi$: an odd natural can be read through action r

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

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

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 $D \to 2^D$ to every predicate variable

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

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

$$[\![\phi_1 \vee \phi_2]\!]\theta\varepsilon = [\![\phi_1]\!]\theta\varepsilon \bigcup [\![\phi_2]\!]\theta\varepsilon$$
$$[\![\phi_1 \wedge \phi_2]\!]\theta\varepsilon = [\![\phi_1]\!]\theta\varepsilon \bigcap [\![\phi_2]\!]\theta\varepsilon$$

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 $D \to 2^D$ to every predicate variable

$$
\begin{aligned}
[\![\forall d{:}D.\ \phi]\!]\theta\varepsilon &= \bigcap_{v \in D} [\![\phi]\!]\, \theta\varepsilon[d := v] \ldots \text{. variable } d \text{ gets value } v \\
[\![\exists d{:}D.\ \phi]\!]\theta\varepsilon &= \bigcup_{v \in D} [\![\phi]\!]\, \theta\varepsilon[d := v] \ldots \text{. variable } d \text{ gets value } v
\end{aligned}
$$

$$
\begin{aligned}
[\![\langle\alpha\rangle\phi]\!]\theta\varepsilon &= \{d \mid \exists d' \in D.\ \exists a \in \mathit{Act}.\ \exists v_a \in D_a. \\
&\qquad d \xrightarrow{a(v_a)} d' \wedge a(v_a) \in [\![\alpha]\!]\varepsilon \wedge d' \in [\![\phi]\!]\theta\varepsilon\} \\
[\![[\alpha]\phi]\!]\theta\varepsilon &= \{d \mid \forall d' \in D.\ \forall a \in \mathit{Act}.\ \forall v_a \in D_a. \\
&\qquad (d \xrightarrow{a(v_a)} d' \wedge a(v_a) \in [\![\alpha]\!]\varepsilon) \Rightarrow d' \in [\![\phi]\!]\theta\varepsilon\}
\end{aligned}
$$

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 θ assigns a function $D \to 2^D$ to every predicate variable

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

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

$$[\![(\nu Z(d{:}D := e).\ \phi)]\!]\theta\varepsilon = \mathrm{gfp}(\Phi_{\theta\varepsilon})([\![e]\!]\varepsilon)$$
$$[\![(\mu Z(d{:}D := e).\ \phi)]\!]\theta\varepsilon = \mathrm{lfp}(\Phi_{\theta\varepsilon})([\![e]\!]\varepsilon)$$

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:
   $\nu X.\ [\neg(\exists m{:}Nat.\ r(m))]X \wedge \forall n{:}Nat.\ [r(n)]\langle s(n)\rangle\text{true}$
3. The values consecutively received over action r are ascending:
   $\nu X(n{:}Nat := 0).[\neg(\exists m{:}Nat.r(m))]X(n) \wedge \forall m{:}Nat.[r(m)](m \geq n \wedge X(m))$
   - any value received is always at least 0 ..... therefore $\nu X(n{:}Nat := 0)$
   - the set of read actions ..... $\exists m{:}Nat.\ r(m)$
   - non-read actions are "ignored" ..... $[\neg(\exists m{:}Nat.r(m))]X(n)$
   - 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

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:

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

- 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)

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

- b is a boolean expression

Example

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

Parameterised Boolean Equation Systems

Semantics (1)

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

$$
\begin{array}{ll}
[\![b]\!]\eta\varepsilon & = \text{true if } \varepsilon(b) \text{ holds, else false} \\
[\![X(e)]\!]\eta\varepsilon & = \eta(X)(\varepsilon(e)) \\
[\![\phi_1 \wedge \phi_2]\!]\eta\varepsilon & = [\![\phi_1]\!] \text{ and } [\![\phi_2]\!] \\
[\![\phi_1 \vee \phi_2]\!]\eta\varepsilon & = [\![\phi_1]\!] \text{ or } [\![\phi_2]\!] \\
[\![\forall d : D.\phi]\!]\eta\varepsilon & = \text{for all } v \in D \; [\![\phi]\!]\eta(\varepsilon[d := v]) \\
[\![\exists d : D.\phi]\!]\eta\varepsilon & = \text{for some } v \in D \; [\![\phi]\!]\eta(\varepsilon[d := v])
\end{array}
$$

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: $\eta : Var \to (D \to Bool)$

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

$$\begin{cases} [\![\epsilon]\!]\eta\varepsilon & := \eta \\ [\![(\mu X(d:D) = \phi)\ \mathcal{E}]\!]\eta\varepsilon & := [\![\mathcal{E}]\!]\eta[X := [\![\phi]\!](\eta_\mu\varepsilon)]\varepsilon \\ [\![(\nu X(d:D) = \phi)\ \mathcal{E}]\!]\eta\varepsilon & := [\![\mathcal{E}]\!]\eta[X := [\![\phi]\!](\eta_\nu\varepsilon)]\varepsilon \end{cases}$$

- $[\![\phi]\!]\eta_\mu\varepsilon := \mathsf{lfp}(\lambda f : D \to Bool.\ \lambda v \in D.\ [\![\phi]\!]\eta[X := f]\varepsilon[d := v])$
- $[\![\phi]\!]\eta_\nu\varepsilon := \mathsf{gfp}(\lambda f : D \to Bool.\ \lambda v \in D.\ [\![\phi]\!]\eta[X := f]\varepsilon[d := v])$

Outline

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:

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

- 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 $\sigma Z \ldots$ 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

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

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

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

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

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

Transforming Satisfiability to Solving PBESs

- Given is a First-order Modal 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

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

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

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

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

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

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

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

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

Transforming Satisfiability to Solving PBESs

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

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

- $\mathsf{match}(a_i(d_{a_i}))$ can always be brought into Positive Normal Form
- Hence, $\mathsf{match}(a_i(d_{a_i}))$ is a predicate formula
- $\mathsf{match}(a_i(d_{a_i}))$ does not introduce predicate variables
- $\mathsf{match}(a_i(d_{a_i}))$ will not cause problems on the left-hand side of an implication

Transforming Satisfiability to Solving PBESs

Example

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

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:

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

Outline

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 $\forall d{:}D.\langle \alpha \rangle \phi$