Alcino Cunha

SPECIFICATION AND MODELING

RELATIONAL LOGIC

Universidade do Minho & INESC TEC

2019/20

RELATIONAL LOGIC

- Adds to first-order logic combinators to build more complex predicates (aka *relational expressions*) out of simpler ones
- Also adds closure operators, technically enhancing the expressive power of first-order logic
- It can considerably simplify the specification of some constraints
- First-order logic specifications tend to be *point-wise* (lots of quantifiers) while relational logic favours a more *point-free* style

SYNTAX

Category	Identifier
…	…
Unary (relational) expressions	$A, B, \ldots$
Higher-arity (relational) expressions	$\Phi, \Psi, \ldots$

SYNTAX

$$
\begin{array}{lll}
\phi & ::= & \Phi(t_1, \dots, t_{\mathrm{ar}(\Phi)}) \\
& \mid & \Phi \subseteq \Psi \qquad\qquad \mathrm{ar}(\Phi) = \mathrm{ar}(\Psi) \\
& \mid & |\Phi| \leq 1 \\
& \mid & |\Phi| \geq 1 \\
& \mid & \dots
\end{array}
$$

SYNTAX

$$
\begin{array}{rcll}
\Phi & ::= & P & \\
 & | & \mathrm{id} & \\
 & | & t & \\
 & | & \Phi_1 \cup \Phi_2 & \mathrm{ar}(\Phi) = \mathrm{ar}(\Psi) \\
 & | & \Phi_1 \cap \Phi_2 & \mathrm{ar}(\Phi) = \mathrm{ar}(\Psi) \\
 & | & \Phi_1 \setminus \Phi_2 & \mathrm{ar}(\Phi) = \mathrm{ar}(\Psi) \\
 & | & \Phi_1 \times \Phi_2 & \\
 & | & \Phi_1 \, . \, \Phi_2 & \mathrm{ar}(\Phi) + \mathrm{ar}(\Psi) > 2 \\
 & | & A \triangleleft \Phi & \\
 & | & \Phi \triangleright A & \\
 & | & \Phi^{\circ} & \mathrm{ar}(\Phi) = 2 \\
 & | & \Phi^{+} & \mathrm{ar}(\Phi) = 2 \\
 & | & \Phi^{\star} & \mathrm{ar}(\Phi) = 2 \\
 & | & \{x_1, \ldots, x_n \mid \phi\} &
\end{array}
$$

SEMANTICS

$$\begin{aligned}
\mathcal{M} &\models \Phi \subseteq \Psi \quad \text{iff} \quad \mathcal{M} \models \forall \vec{x} \cdot \Phi(\vec{x}) \rightarrow \Psi(\vec{x}) \\
\mathcal{M} &\models |\Phi| \leq 1 \quad \text{iff} \quad \mathcal{M} \models \forall \vec{x} \cdot \forall \vec{y} \cdot \Phi(\vec{x}) \wedge \Phi(\vec{y}) \rightarrow \vec{x} = \vec{y} \\
\mathcal{M} &\models |\Phi| \geq 1 \quad \text{iff} \quad \mathcal{M} \models \exists \vec{x} \cdot \Phi(\vec{x})
\end{aligned}$$

SEMANTICS

$$
\begin{aligned}
\mathcal{M} &\models P(\vec{t}) &\text{iff}\quad & (\mathcal{M}(t_1), \dots, \mathcal{M}(t_{\mathrm{ar}(P)})) \in \mathcal{M}(P) \\
\mathcal{M} &\models \mathrm{id}(t, u) &\text{iff}\quad & \mathcal{M}(t) = \mathcal{M}(u) \\
\mathcal{M} &\models t(u) &\text{iff}\quad & \mathcal{M}(t) = \mathcal{M}(u) \\
\mathcal{M} &\models (\Phi \cup \Psi)(\vec{t}) &\text{iff}\quad & \mathcal{M} \models \Phi(\vec{t}) \vee \Psi(\vec{t}) \\
\mathcal{M} &\models (\Phi \cap \Psi)(\vec{t}) &\text{iff}\quad & \mathcal{M} \models \Phi(\vec{t}) \wedge \Psi(\vec{t}) \\
\mathcal{M} &\models (\Phi \setminus \Psi)(\vec{t}) &\text{iff}\quad & \mathcal{M} \models \Phi(\vec{t}) \wedge \neg\Psi(\vec{t}) \\
\mathcal{M} &\models (\Phi \times \Psi)(\vec{t}) &\text{iff}\quad & \mathcal{M} \models \Phi(t_1, \dots, t_{\mathrm{ar}(\Phi)}) \wedge \Psi(t_{\mathrm{ar}(\Phi)+1}, \dots, t_{\mathrm{ar}(\Phi)+\mathrm{ar}(\Psi)}) \\
\mathcal{M} &\models (\Phi \,.\, \Psi)(\vec{t}) &\text{iff}\quad & \mathcal{M} \models \exists\, x \cdot \Phi(t_1, \dots, t_{\mathrm{ar}(\Phi)-1}, x) \wedge \Psi(x, t_{\mathrm{ar}(\Phi)}, \dots, t_{\mathrm{ar}(\Phi)+\mathrm{ar}(\Psi)} \\
\mathcal{M} &\models (A \lhd \Phi)(\vec{t}) &\text{iff}\quad & \mathcal{M} \models A(t_1) \wedge \Phi(\vec{t}) \\
\mathcal{M} &\models (\Phi \rhd A)(\vec{t}) &\text{iff}\quad & \mathcal{M} \models \Phi(\vec{t}) \wedge A(t_{\mathrm{ar}(\Phi)}) \\
\mathcal{M} &\models \Phi^{\circ}(t, u) &\text{iff}\quad & \mathcal{M} \models \Phi(u, t) \\
\mathcal{M} &\models \Phi^{+}(t, u) &\text{iff}\quad & \mathcal{M} \models \Phi(t, u) \vee \exists\, x \cdot \Phi(t, x) \wedge \Phi^{+}(x, u) \\
\mathcal{M} &\models \Phi^{\star}(t, u) &\text{iff}\quad & \mathcal{M} \models \Phi^{+}(t, u) \vee t = u \\
\mathcal{M} &\models \{x_1, \dots, x_n \mid \phi\}(\vec{t}) &\text{iff}\quad & \phi[t_1/x_1, \dots, t_n/x_n]
\end{aligned}
$$

SEMANTICS

- It is more intuitive to assume the existence of a function $[\![\cdot]\!]_{\mathcal{M}}$ that computes the value of a relational expression in a model

$$\mathcal{M} \models \Phi(t_1, \dots, t_n) \quad \text{iff} \quad (\mathcal{M}(t_1), \dots, \mathcal{M}(t_n)) \in [\![\Phi]\!]_{\mathcal{M}}$$

- The following slides explain this function by example

SEMANTICS BY EXAMPLE

$$
\begin{aligned}
\mathcal{M}(\texttt{Prepared}) &= \{(\texttt{W1}), (\texttt{W2})\} \\
\mathcal{M}(\texttt{Aborted}) &= \{(\texttt{W2}), (\texttt{W3})\} \\
\mathcal{M}(\texttt{Task}) &= \{(\texttt{T1}), (\texttt{T2})\} \\
\mathcal{M}(\texttt{working\_on}) &= \{(\texttt{W1}, \texttt{T1}), (\texttt{W2}, \texttt{T2}), (\texttt{W2}, \texttt{T3}), (\texttt{W3}, \texttt{T3})\}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{Prepared} \cup \texttt{Aborted}]\!]_{\mathcal{M}} &= \{(\texttt{W1}), (\texttt{W2}), (\texttt{W3})\} \\
[\![\texttt{Prepared} \cap \texttt{Aborted}]\!]_{\mathcal{M}} &= \{(\texttt{W2})\} \\
[\![\texttt{Prepared} \setminus \texttt{Aborted}]\!]_{\mathcal{M}} &= \{(\texttt{W1})\} \\
[\![\texttt{W1} \times \texttt{Task}]\!]_{\mathcal{M}} &= \{(\texttt{W1}, \texttt{T1}), (\texttt{W1}, \texttt{T2})\} \\
[\![\texttt{Prepared} \lhd \texttt{working\_on}]\!]_{\mathcal{M}} &= \{(\texttt{W1}, \texttt{T1}), (\texttt{W2}, \texttt{T2}), (\texttt{W2}, \texttt{T3})\} \\
[\![\texttt{working\_on} \rhd \texttt{Task}]\!]_{\mathcal{M}} &= \{(\texttt{W1}, \texttt{T1}), (\texttt{W2}, \texttt{T2})\} \\
[\![\texttt{working\_on}^{\circ}]\!]_{\mathcal{M}} &= \{(\texttt{T1}, \texttt{W1}), (\texttt{T2}, \texttt{W2}), (\texttt{T3}, \texttt{W2}), (\texttt{T3}, \texttt{W3})\} \\
[\![\{x, y \mid x = y \land \texttt{Prepared}(x)\}]\!]_{\mathcal{M}} &= \{(\texttt{W1}, \texttt{W1}), (\texttt{W2}, \texttt{W2})\}
\end{aligned}
$$

SEMANTICS BY EXAMPLE

$$
\begin{aligned}
\mathcal{M}(\texttt{Prepared}) &= \{(\texttt{W1}), (\texttt{W2})\} \\
\mathcal{M}(\texttt{working\_on}) &= \{(\texttt{W1}, \texttt{T1}), (\texttt{W2}, \texttt{T1}), (\texttt{W2}, \texttt{T2})\} \\
\mathcal{M}(\texttt{requires}) &= \{(\texttt{T1}, \texttt{T2}), (\texttt{T2}, \texttt{T3}), (\texttt{T3}, \texttt{T4})\}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{W1 . working\_on}]\!]_{\mathcal{M}} &= \{(\texttt{T1})\} \\
[\![\texttt{working\_on . T1}]\!]_{\mathcal{M}} &= \{(\texttt{W1}), (\texttt{W2})\} \\
[\![\texttt{Prepared . working\_on}]\!]_{\mathcal{M}} &= \{(\texttt{T1}), (\texttt{T2})\} \\
[\![\texttt{working\_on . requires}]\!]_{\mathcal{M}} &= \{(\texttt{W1}, \texttt{T2}), (\texttt{W2}, \texttt{T2}), (\texttt{W2}, \texttt{T3})\} \\
[\![\texttt{working\_on . working\_on}^{\circ}]\!]_{\mathcal{M}} &= \{(\texttt{W1}, \texttt{W1}), (\texttt{W1}, \texttt{W2}), (\texttt{W2}, \texttt{W1}), (\texttt{W2}, \texttt{W2})\} \\
[\![\texttt{W1 . working\_on . requires}^{+}]\!]_{\mathcal{M}} &= \{(\texttt{T2}), (\texttt{T3}), (\texttt{T4})\}
\end{aligned}
$$

RELATIONAL LOGIC SYNTAX IN ALLOY

Alloy	Math
Φ **in** Ψ	$\Phi \subseteq \Psi$
Φ = Ψ	$\Phi = \Psi$
lone Φ	$\lvert\Phi\rvert \leq 1$
some Φ	$\lvert\Phi\rvert \geq 1$
no Φ	$\lvert\Phi\rvert = 0$
one Φ	$\lvert\Phi\rvert = 1$

RELATIONAL LOGIC SYNTAX IN ALLOY

Alloy	Math
iden	id
`Φ + Ψ`	$\Phi \cup \Psi$
`Φ & Ψ`	$\Phi \cap \Psi$
`Φ - Ψ`	$\Phi \setminus \Psi$
`Φ -> Ψ`	$\Phi \times \Psi$
`Φ . Ψ`	$\Phi \,.\, \Psi$
`A <: Φ`	$A \lhd \Psi$
`Φ :> A`	$\Phi \rhd A$
`~ Φ`	Φ°
`^ Φ`	Φ^{+}
`* Φ`	$\Phi^{\star}$
`{ x : A | ϕ }`	$\{x \mid x \in A \wedge \phi\}$

FORMULA EXAMPLES

```
sig Worker {
  working_on : set Task,
}
sig Prepared in Worker {}
sig Committed in Prepared {}
sig Aborted in Worker {}
sig Task {
  requires : set Task
}
```

FORMULA EXAMPLES

```
-- There are no prepared workers
no Prepared
-- Every worker is either committed or aborted
Worker = Committed + Aborted
no Committed & Aborted
-- No worker is working on a task
no working_on
-- Every worker is working on at least one task
all w : Worker | some w.working_on
-- Every worker is working on at most one task
all w : Worker | lone w.working_on
-- Dependency between tasks is acyclic
all t : Task | t not in t.^requires
```

A QUESTION OF STYLE

```
-- No shared tasks, point-wise style
all x, y : Worker, t : Task | x->t in working_on and y->t in working_on
                              implies x = y

-- No shared tasks, navigational (mixed) style
all t : Task | lone working_on.t

-- No shared tasks, point-free style
working_on.~working_on in iden
```