Alcino Cunha

SPECIFICATION AND MODELING

FIRST-ORDER LOGIC

Universidade do Minho & INESC TEC

2019/20

FROM PROPOSITIONAL TO FIRST-ORDER LOGIC

- Introduces a *domain* or *universe* of discourse
- Generalize propositional symbols to *predicates*
- Allows *quantifiers* and *variables* ranging over the domain

Propositional logic

$\texttt{Worker1\_Prepared} \land \texttt{Worker2\_Prepared}$

$\texttt{Worker2\_working\_on\_Task1}$

First-order logic

$\texttt{Prepared}(\texttt{Worker1}) \land \texttt{Prepared}(\texttt{Worker2})$

$\texttt{working\_on}(\texttt{Worker2}, \texttt{Task1})$

$\forall x.\texttt{Prepared}(x)$

PREDICATES (AKA SETS AND RELATIONS)

$$
\begin{aligned}
\texttt{Prepared}(\texttt{Worker1}) &= \top \\
\texttt{Prepared}(\texttt{Worker2}) &= \top \\
\texttt{Prepared}(\_) &= \bot \\
\texttt{working\_on}(\texttt{Worker1}, \texttt{Task1}) &= \top \\
\texttt{working\_on}(\texttt{Worker1}, \texttt{Task2}) &= \top \\
\texttt{working\_on}(\texttt{Worker2}, \texttt{Task3}) &= \top \\
\texttt{working\_on}(\_, \_) &= \bot
\end{aligned}
$$

$$
\begin{aligned}
\texttt{Prepared} &= \{(\texttt{Worker1}), (\texttt{Worker2})\} \\
\texttt{working\_on} &= \{(\texttt{Worker1}, \texttt{Task1}), (\texttt{Worker1}, \texttt{Task2}), (\texttt{Worker2}, \texttt{Task3})\}
\end{aligned}
$$

SYNTAX

Category	Identifier
Variables	$x, y, z, \ldots$
Constants	$a, b, c, \ldots$
Predicates	$P, Q, R, \ldots$
Terms	$t, u, v, \ldots$
Formulas	$\phi, \varphi, \psi, \ldots$

SYNTAX

$$
\begin{array}{rcl}
t & ::= & x \\
  & | & c \\
  \\
\phi & ::= & P(t_1, \ldots, t_{\mathrm{ar}(P)}) \\
  & | & t = u \\
  & | & \top \\
  & | & \bot \\
  & | & \neg\phi \\
  & | & \phi_1 \wedge \phi_2 \\
  & | & \phi_1 \vee \phi_2 \\
  & | & \phi_1 \rightarrow \phi_2 \\
  & | & \forall x \cdot \phi \\
  & | & \exists x \cdot \phi
\end{array}
$$

FIRST-ORDER STRUCTURES AND VARIABLE ASSIGNMENTS

- The semantics of a first-order formula is defined over a *first-order structure* $\mathcal{U}, \mathcal{M}$ where:
  - $\mathcal{U}$ is a non-empty domain (or *universe*) of interpretation with equality
  - $\mathcal{M}$ is an interpretation (*model*) for constants, predicates, and variables:
    - $\mathcal{M}(c) \in \mathcal{U}$
    - $\mathcal{M}(x) \in \mathcal{U}$
    - $\mathcal{M}(P) \subseteq \mathcal{U}^{\mathrm{ar}(P)}$
- The fact that a formula ϕ is valid in a model $\mathcal{M}$ with universe $\mathcal{U}$ is denoted by $\mathcal{U}, \mathcal{M} \models \phi$
- When $\mathcal{U}$ is implicit or clear from context we write just $\mathcal{M} \models \phi$

EXAMPLE

- Assuming:
  - $\mathcal{U} = \{\texttt{Worker1}, \texttt{Worker2}, \texttt{Task1}, \texttt{Task2}\}$
  - $\mathcal{M}(\texttt{Worker}) = \{(\texttt{Worker1}), (\texttt{Worker2})\}$
  - $\mathcal{M}(\texttt{Task}) = \{(\texttt{Task1}), (\texttt{Task2})\}$
  - $\mathcal{M}(\texttt{Prepared}) = \{(\texttt{Worker1}), (\texttt{Worker2})\}$
  - $\mathcal{M}(\texttt{working\_on}) = \{(\texttt{Worker2}, \texttt{Task1})\}$
- We have:

$$\mathcal{M} \models \forall x \cdot \texttt{Worker}(x) \vee \texttt{Task}(x)$$
$$\mathcal{M} \models \forall x \cdot \texttt{Worker}(x) \longrightarrow \texttt{Prepared}(x)$$
$$\mathcal{M} \not\models \forall x \cdot \texttt{Worker}(x) \longrightarrow \exists y \cdot \texttt{Task}(y) \wedge \texttt{working\_on}(x, y)$$

SEMANTICS

$$
\begin{aligned}
\mathcal{M} \models P(t_1, \dots, t_n) &\quad \text{iff} \quad (\mathcal{M}(t_1), \dots, \mathcal{M}(t_n) \in \mathcal{M}(P) \\
\mathcal{M} \models t = u &\quad \text{iff} \quad \mathcal{M}(t) = \mathcal{M}(u) \\
\mathcal{M} \models \top & \\
\mathcal{M} \not\models \bot & \\
\mathcal{M} \models \neg\phi &\quad \text{iff} \quad \mathcal{M} \not\models \phi \\
\mathcal{M} \models \phi_1 \wedge \phi_2 &\quad \text{iff} \quad \mathcal{M} \models \phi_1 \text{ and } \mathcal{M} \models \phi_2 \\
\mathcal{M} \models \phi_1 \vee \phi_2 &\quad \text{iff} \quad \mathcal{M} \models \phi_1 \text{ or } \mathcal{M} \models \phi_2 \\
\mathcal{M} \models \phi_1 \longrightarrow \phi_2 &\quad \text{iff} \quad \mathcal{M} \not\models \phi \text{ or } \mathcal{M} \models \phi_2 \\
\mathcal{M} \models \forall x \cdot \phi &\quad \text{iff} \quad \mathcal{M}[x \mapsto a] \models \phi \text{ for all } a \in \mathcal{U} \\
\mathcal{M} \models \exists x \cdot \phi &\quad \text{iff} \quad \mathcal{M}[x \mapsto a] \models \phi \text{ for some } a \in \mathcal{U}
\end{aligned}
$$

FIRST-ORDER LOGIC IN ALLOY

- The universe is a set of uninterpreted *atoms*
- No constants and no functions
- No false nor true
- Quantifications always range over a unary predicate

$$
\begin{array}{rcl}
\phi & ::= & P(x_1, \ldots, x_{\mathrm{ar}(P)}) \\
 & \mid & x = y \\
 & \mid & \neg\phi \\
 & \mid & \phi_1 \land \phi_2 \\
 & \mid & \phi_1 \lor \phi_2 \\
 & \mid & \phi_1 \longrightarrow \phi_2 \\
 & \mid & \forall x \cdot P(x) \longrightarrow \phi \\
 & \mid & \exists x \cdot P(x) \land \phi
\end{array}
$$

FIRST-ORDER LOGIC SYNTAX IN ALLOY

Alloy	Math
x_1 `->` … `->` x_n **in** P	$P(x_1, \ldots, x_n)$
x_1 `->` … `->` x_n **not in** P	$\neg P(x_1, \ldots, x_n)$
x `=` y	$x = y$
x `!=` y	$\neg(x = y)$
not ϕ	$\neg \phi$
ϕ_1 **and** ϕ_2	$\phi_1 \wedge \phi_2$
ϕ_1 **or** ϕ_2	$\phi_1 \vee \phi_2$
ϕ_1 **implies** ϕ_2	$\phi_1 \longrightarrow \phi_2$
all x : P | ϕ	$\forall x \cdot P(x) \longrightarrow \phi$
some x : P | ϕ	$\exists x \cdot P(x) \wedge \phi$

PREDICATE DECLARATIONS IN ALLOY

- Unary predicates are known as *signatures* or *sets*
  - Declared with the **sig** keyword
  - Sub-set signatures are declared with the **in** keyword
- Predicates of higher arity are known as *relations*
  - Declared inside signatures

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

PREDICATE DECLARATIONS IN ALLOY

- Declarations induce a set of implicit “typing” constraints
  - *Top-level* (non sub-set) signatures are disjoint
  - Sub-set signatures are indeed sub-sets of the parent signature
  - Relations only contain tuples of the correct signatures
- Some special predicates are pre-defined
  - **`univ`** is the union of all top-level signatures
  - **`none`** is the empty set
  - **`iden`** is the identity binary relation over **`univ`**

FORMULA EXAMPLES

```
-- There are no prepared workers
all w : Worker | w not in Prepared
-- Every worker is either committed or aborted
all w : Worker | w in Committed or w in Aborted
all w : Worker | w in Committed implies w not in Aborted
-- No worker is working on a task
all w : Worker | all t : Task | w->t not in working_on
-- Every worker is working on at least one task
all w : Worker | some t : Task | w->t in working_on
-- Every worker is working on at most one task
all w : Worker, t,u : Task |
    w->t in working_on and w->u in working_on implies t=u
```

WHAT ABOUT SET INCLUSION AND SET OPERATORS?

- Set inclusion can be defined in first-order logic

$$A \subseteq B \equiv \forall x \cdot A(x) \longrightarrow B(x)$$

- Set operators act like combinators that build more complex (unary) predicates out of simpler ones

$$(A \cup B)(x) \equiv A(x) \vee B(x)$$

- These (and other) combinators simplify the specification of constraints
- They will be the subject of our next class about *relational logic*, the logic of Alloy!