CS:5810
Formal Methods in Software Engineering

Dynamic Models in Alloy
Overview

• Basics of dynamic models
  – Modeling a system’s states and state transitions
  – Modeling operations causing transitions

• Simple example of operations
Static Models

- So far we’ve used Alloy to define the allowable values of state components
  - values of sets
  - values of relations
- A model instance is a set of state component values that
  - Satisfies the constraints defined by multiplicities, fact, “realism” conditions, ...
Static Models

Person = {Matt, Sue}
Man = {Matt}
Woman = {Sue}
Married = {Matt, Sue}
spouse = {Matt, Sue}
children = {}
siblings = {}

Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
Married = {Matt, Sue}
spouse = {Matt, Sue}
children = {(Matt, Sean), (Sue, Sean)}
siblings = {}
Dynamic Models

• Static models allow us to describe the legal states of a dynamic system

• We also want to be able to describe the legal transitions between states

  E.g.
  – To get married one must be alive and not currently married
  – One must be alive to be able to die
  – A person becomes someone’s child after birth
Example

Family Model

```java
abstract sig Person {
    children: set Person,
    siblings: set Person
}

sig Man, Woman extends Person {}

sig Married in Person {
    spouse: one Married
}
```
State Transitions

• Two people get married

  – At time t, spouse = {}
  – At time t’, spouse = {(Matt, Sue), (Sue, Matt)}

⇒ We add the notion of time in the relation spouse
Modeling State Transitions

• Alloy has no predefined notion of state transition

• However, there are several ways to model dynamic aspects of a system in Alloy

• A general and relatively simple way is to:
  1. introduce a Time signature expressing time
  2. add a time component to each relation that changes over time
Family Model Signatures

\[
\text{abstract sig Person } \{
\begin{align*}
\text{children: set Person,} \\
\text{siblings: set Person set}
\end{align*}
\}
\]

\[
\text{sig Man, Woman extends Person } \{
\}
\]

\[
\text{sig Married in Person } \{
\begin{align*}
\text{spouse: one Married one}
\end{align*}
\}
\]
Family Model Signatures with Time

\texttt{sig Time }

\texttt{abstract sig Person }
\begin{itemize}
\item \texttt{children: Person set \rightarrow Time,}
\item \texttt{siblings: Person set \rightarrow Time}
\end{itemize}

\texttt{sig Man, Woman extends Person }

\texttt{sig Married in Person }
\begin{itemize}
\item \texttt{spouse: Married one \rightarrow Time}
\end{itemize}
Transitions

• Two people get married

  – At time $t$, $\text{Married} = \{\}$
  – At time $t'$, $\text{Married} = \{\text{Matt, Sue}\}$

  – Actually, we can’t have a time-dependent signature such as $\text{Married}$ because signatures are not time dependent
Transitions

• A person is born
  – At time $t$, Person = {}
  – At time $t'$, Person = {Sue}
  – We cannot add the notion being born to the signature Person because signatures are not time dependent
Signatures are Static

abstract sig Person {
  children: Person set -> Time,
  siblings: Person set -> Time,
  spouse: Person lone -> Time
}
sig Man, Woman extends Person {}

sig Married in Person {
  spouse: Married one -> Time
}
Signatures are Static

abstract sig Person {
    children: Person set -> Time,
    siblings: Person set -> Time,
    spouse: Person lone -> Time
    alive: set Time
}

sig Man, Woman extends Person {}
Revising Constraints

abstract sig Person {
  children: Person set -> Time,
  siblings: Person set -> Time,
  spouse: Person lone -> Time,
  alive: set Time
}
Revising Constraints

abstract sig Person {
  children: Person set -> Time,
  siblings: Person set -> Time,
  spouse: Person lone -> Time,
  alive: set Time
  parents: Person set -> Time
}

sig Man, Woman extends Person {}

fun parents[] : Person -> Person {~children}

fact parentsDef {
  all t: Time | parents.t = ~(children.t)
}
Revising Constraints

-- Time-dependent parents relation

fact parentsDef {  
    all t: Time | parents.t = ~(children.t)  
}

-- Two persons are blood relatives iff
-- they have a common ancestor
pred BloodRelatives [p, q: Person, t: Time] {  
    some p.*(parents.t) & q.*(parents.t)  
}
Revising Static Constraints

-- People cannot be their own ancestors
\textbf{all} \ t: \ Time \ | \ \textbf{no} \ p: \ Person \ |
\hfill p \ \textbf{in} \ p.\wedge(parents.\ t)

-- No one can have more than one father
-- or mother
\textbf{all} \ t: \ Time \ | \ \textbf{all} \ p: \ Person \ |
\textbf{lone} \ (p.\text{parents}.\ t \ & \ \text{Man})
\textbf{and}
\textbf{lone} \ (p.\text{parents}.\ t \ & \ \text{Woman})

...
Revising Static Constraints

-- A person p's siblings are those people, other than p, with the same parents as p

\[
\text{all } t: \text{Time} \mid \text{all } p: \text{Person} \mid \\
\quad p.\text{siblings}.t = \\
\quad \{ q: \text{Person} - p \mid \text{some } q.\text{parents}.t \text{ and } \\
\quad \quad p.\text{parents}.t = q.\text{parents}.t \} 
\]

-- Each married man (woman) has a wife (husband)

\[
\text{all } t: \text{Time} \mid \text{all } p: \text{Person} \mid \\
\quad \text{let } s = p.\text{spouse}.t \mid \\
\quad (p \text{ in Man implies } s \text{ in Woman}) \text{ and } \\
\quad (p \text{ in Woman implies } s \text{ in Man})
\]
Revising *Static* Constraints

-- A spouse can't be a sibling

\[
\text{all } t: \text{Time} \mid \text{no } p: \text{Person} \mid \\
\text{some } p.\text{spouse}.t \text{ and } \\
p.\text{spouse}.t \text{ in } p.\text{siblings}.t
\]

-- People can't be married to a blood relative

\[
\text{all } t: \text{Time} \mid \text{no } p: \text{Person} \mid \\
\text{let } s = p.\text{spouse}.t \mid \\
\text{some } s \text{ and } \\
\text{BloodRelatives}[p, s, t]
\]
Revising *Static* Constraints

-- a person can't have children with
-- a blood relative

\[
\text{all } t : \text{Time} \mid \text{all } p, q : \text{Person} \mid
(some (p.children.t \& q.children.t) \text{ and } p \neq q)
\implies
\text{not BloodRelatives}[p, q, t]
\]

-- the spouse relation is symmetric

\[
\text{all } t : \text{Time} \mid
\text{spouse.t} = \sim(\text{spouse.t})
\]
Exercises

• Load family-6.als
• Execute it
• Analyze the model
• Look at the generated instance
• Does it look correct?
• What, if anything, would you change about it?
Transitions

A person is born from parents

- Add to **alive** relation
- Modify children/parents relations
State Sequences

Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
spouse = {}
children = {}
siblings = {}
alive = {Sue}

Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
spouse = {{Matt, Sue}, (Sue, Matt)}
children = {}
siblings = {}
alive = {Sue, Matt}

Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
spouse = {{Matt, Sue}, (Sue, Matt)}
children = {{Matt, Sean}, (Sue, Sean)}
siblings = {}alive = {Sue, Matt, Sean}

CS:5810 -- Formal Methods in Software Engineering Fall 2018
Expressing Transitions in Alloy

• A transition can be thought of as caused by the application of an operator to the current state

• An operator can be modeled as a predicate over two states:
  1. the state right before the transition and
  2. the state right after it

• We define it as predicate with (at least) two formal parameters: $t, t'$: Time

• Constraints over time $t$ (resp., $t'$) model the state right before (resp., after) the transition
Expressing Transitions in Alloy

• Pre-condition constraints
  – Describe the states to which the transition applies

• Post-condition constraints
  – Describes the effects of the transition in generating the next state

• Frame-condition constraints
  – Describes what does not change between pre-state and post-state of a transition

*Distinguishing the pre-, post- and frame-conditions in comments provides useful documentation*
Example: Marriage

pred getMarried [m: Man, w: Woman, t,t': Time] {
  -- preconditions
  -- m and w must be alive
  m+w in alive.t
  -- neither one is married
  no (m+w).spouse.t
  -- they are not be blood relatives
  not BloodRelatives[m, w, t]
  -- post-conditions
  -- w is m’s wife
  m.spouse.t' = w
  -- m is w’s husband
  w.spouse.t' = m
  -- frame conditions ??
}
Frame Condition

How is each relation touched by marriage?

• 5 relations:
  – children, parents, siblings
  – spouse
  – alive

• parents and siblings relations are defined in terms of the children relation

• Thus, the frame condition has only to consider children, spouse and alive relations
Frame Condition Predicates

\textbf{pred} \ noChildrenChangeExcept [ps: set Person t,t': Time] {
\begin{align*}
\text{all} & \ p: \text{Person} - ps | \\
& \ \ \ \ p.\text{children}.t' = p.\text{children}.t \\
\end{align*}
}

\textbf{pred} \ noSpouseChangeExcept [ps: set Person t,t': Time] {
\begin{align*}
\text{all} & \ p: \text{Person} - ps | \\
& \ \ \ \ p.\text{spouse}.t' = p.\text{spouse}.t \\
\end{align*}
}

\textbf{pred} \ noAliveChange [t,t': Time] {
\begin{align*}
\text{alive}.t' = \text{alive}.t \\
\end{align*}
}
Example: Marriage

pred getMarried[\text{m: Man, w: Woman, t,t': Time}] 
{
  -- preconditions
  \text{m+w in alive.t}
  \text{no (m+w).spouse.t}
  \text{not BloodRelatives[m, w, t]}
  -- post-conditions
  \text{m.spouse.t' = w}
  -- frame conditions
  \text{noSpouseChangeExcept[m+w, t, t']}
  \text{noChildrenChangeExcept[none, t, t']}
  \text{noAliveChange[t, t']}
}
Instance of Marriage

```plaintext
open ordering [Time] as T

pred marriageInstance {
  some t: Time |
  some m: Man | some w: Woman |
  getMarried[m, w, t, T/next[t] ]
}
run { marriageInstance }
```
Example: Birth from Parents

```plaintext
pred isBornFromParents [p: Person, m,w: Person, t,t': Time] {

  -- Pre-condition
  m+w in alive.t
  p !in alive.t

  -- Post-condition and frame condition
  alive.t' = alive.t + p
  m.children.t' = m.children.t + p
  w.children.t' = w.children.t + p

  -- Frame condition
  noChildrenChangeExcept[m+w, t, t']
  noSpouseChangeExcept[none, t, t']
}
```
Instance of Birth

\[
\text{pred birthInstance} \{ \\
\quad \text{some } t: \text{Time} \mid \\
\quad \text{some } p1, p2, p3: \text{Person} \mid \\
\quad \quad \text{isBornFromParents}[p1, p2, p3, t, T/\text{next}[t]] \\
\}
\]

\[
\text{run } \{ \text{birthInstance} \}
\]
Example: Death

```plaintext
pred dies [p: Person, t,t': Time] {  
  -- Pre-condition  
  p in alive.t  
  
  -- Post-condition  
  no p.spouse.t'  
  
  -- Post-condition and frame condition  
  alive.t' = alive.t - p  
  all s: p.spouse.t |  
  s.spouse.t' = s.spouse.t - p  
  
  -- Frame condition  
  noChildrenChangeExcept[none, t, t']  
  noSpouseChangeExcept[p + p.spouse.t, t, t']
}
```
Instance of Death

pred deathInstance {
    some t: Time |
    some p: Person |
        dies[p, t, T/next[t]]
}

run { deathInstance }
Specifying Transition Systems

• A transition system can be defined as a set of executions:
  sequences of time steps generated by the operators

• In our example, for every execution:
  – The first time step satisfies some initialization condition
  – Each pair of consecutive steps are related by
    • a birth operation, or
    • a death operation, or
    • a marriage operation
Initial State Specification

`init` specifies constraints on the initial state

```
pred init [t: Time] {  
  no children.t  
  no spouse.t  
  #alive.t > 2  
  #Person > #alive.t  
}
```
Transition Relation Specification

trans specifies that each transition is a consequence of the application of one of the operators to some individuals

\[\text{pred trans [t,t': Time] \{} \]
\[\text{(some m: Man, w: Woman | getMarried [m, w, t, t']}) \]
\[\text{or} \]
\[\text{(some p: Person, m: Man, w: Woman | isBornFromParents [p, m, w, t, t']}) \]
\[\text{or} \]
\[\text{(some p: Person | dies [p, t, t']}) \]
\[\text{}\}\]
System Specification

System specifies that each execution of the system starts in a state satisfying the initial state condition and moves from one state to the next through the application of one operator at a time, until it reaches the final state.

\[
\text{pred System \{} \\
\quad \text{init}[T/\text{first}] \\
\quad \text{all } t: \text{Time} - T/\text{last} | \text{trans}[t, T/next[t]] \\
\}\n\]

\[
\text{run \{} \text{System } \}\n\]
System Invariants

• Many of the facts that we stated in our static model now become expected system invariants

• These are properties that
  – should hold in initial states
  – should be preserved by system transitions

• In Alloy we can check that a property is invariant (in a given scope) by
  – encoding it as a formula $P$ and checking
  – checking the assertion

  \[ \text{System} \Rightarrow \forall t: \text{Time} \mid P \]
Expected Invariants: Examples

-- People cannot be their own ancestors

\[
\text{assert } a1 \{ \text{System } \Rightarrow \text{all } t: \text{Time } | \no p: \text{Person } | p \text{ in } p.\text{^(parents.t)} \}
\]

check a1 for 8

-- No one can have more than one father or mother

\[
\text{assert } a2 \{ \text{System } \Rightarrow \text{all } t: \text{Time } | \all p: \text{Person } | \lone (p.\text{parents.t } \& \text{ Man}) \text{ and } \lone (p.\text{parents.t } \& \text{ Woman}) \}
\]

check a2 for 8
Exercises

• Load family-7.als
• Execute it
• Look at the generated instance
• Does it look correct?
• What if anything would you change about it?
• Check each of the given assertions
• Are they all valid?
• If not, how would you change the model to fix that?