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 = \{\}
marriedWith = \{\}
children = \{\}
sibling = \{\}

Person = \{Matt, Sue, Sean\}
Man = \{Matt\}
Woman = \{Sue\}
Married = \{Matt, Sue\}
marriedWith = \{(Matt,Sue), (Sue,Matt)\}
children = \{(Matt,Sean), (Sue,Sean)\}
sibling = \{\}
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.
  - A person is born before they can be married.
  - A person gets married before having children.
  - A person is a child until they die.
Example

**Family Model**

```plaintext
abstract sig Person {
    children: set Person,
    sibling: set Person
}

sig Man, Woman extends Person {}

sig Married in Person {
    marriedWith: one Married
}
```
Transitions

- Two people get married
  - At time $t$, $\text{marriedWith} = \{\}$
  - At time $t'$, $\text{marriedWith} = \{(\text{Matt}, \text{Sue}), (\text{Sue}, \text{Matt})\}$

$\Rightarrow$ We add the notion of time in the relation $\text{marriedWith}$
Modeling State Transitions

- Alloy does not have an embedded notions of state transition.
- However, there are several ways to model dynamic aspects of a system.
- A general and relatively simple one is to:
  - introduce a **Time** signature expressing time and
  - add a time component to each relation that changes over time.
Summarizing

Family Model

abstract sig Person {
    children: set Person,
    sibling: set Person
}
sig Man, Woman extends Person {}

sig Married in Person {
    marriedWith: one Married
}
Example

Family Model

sig Time {}

abstract sig Person {
  children: Person set -> Time,
  sibling: Person set -> Time
}
sig Man, Woman extends Person {}

sig Married in Person {
  marriedWith: Married one -> Time
}
Transitions

- **Two people get married**
  - At time $t$, Married = {}
  - At time $t'$, Married = {Matt, Sue}
  - Actually, we can’t have a time-dependent signature such as Married because a signature is time independent.

<table>
<thead>
<tr>
<th>Time $t$</th>
<th>Time $t'$</th>
</tr>
</thead>
</table>

Person = {Matt, Sue}
Man = {Matt}
Woman = {Sue}
Married = {}
marrriedWith = {}
children = {}
sibling = {}

Person = {Matt, Sue}
Man = {Matt}
Woman = {Sue}
Married = {Matt, Sue}
marrriedWith = {(Matt,Sue), (Sue,Matt)}
children = {}
sibling = {}
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 a signature is time independent.

Person = {}
Man = {}
Woman = {}
marrriedWith = {}
children = {}
sibling = {}

Time $t$

Person = {Sue}
Man = {}
Woman = {Sue}
marrriedWith = {}
children = {}
sibling = {}

Time $t'$
Signatures are static

**Family Model**

\[
\begin{align*}
\text{abstract sig Person} & \{ \\
\text{children: Person set } & \rightarrow \text{ Time,} \\
\text{sibling: Person set } & \rightarrow \text{ Time,} \\
\text{marriedWith: Person lone } & \rightarrow \text{ Time} \\
\} \\
\text{sig Man, Woman} & \text{ extends Person } \{} \\
\text{sig Married in Person} & \{ \\
\text{marriedWith: Married one } & \rightarrow \text{ Time} \\
\}
\end{align*}
\]
Signatures are static

**Family Model**

```
abstract sig Person {
    children: Person set -> Time,
    sibling: Person set -> Time,
    marriedWith: Person lone -> Time
}
sig Man, Woman extends Person {}
```

We want to add this relation, but where?
```
alive: Person set -> Time
```
Signatures are static

*Family Model*

```plaintext
abstract sig Person {
  children: Person set \rightarrow Time,
  sibling: Person set \rightarrow Time,
  marriedWith: Person lone \rightarrow Time
  alive: set Time
}

sig Man, Woman extends Person {}
```
Revising constraints

Family Model

abstract sig Person {
  children: Person set -> Time,
  sibling: Person set -> Time,
  marriedWith: Person lone -> Time,
  alive: set Time
  parent: Person set -> Time
}

sig Man, Woman extends Person {}

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

fact parentDef {
  all t: Time | all p, q: Person |
  p->q in parent.t <-> q->p in children.t
}
Revising constraints

-- Time-dependent parent relation

```
fact parentDef {
    all t: Time | all p, q: Person |
    p->q in parent.t <=>
    q->p in children.t
}
```

-- Two persons are blood relatives iff
-- they have a common ancestor

```
pred BloodRelatives [p, q: Person, t: Time] {
    some p.*(parent.t) & q.*(parent.t)
}
```
Revising static constraints

```plaintext
fact static {
    -- People cannot be their own ancestors
    all t: Time | no p: Person |
    p in p.^{(parent.t)}

    -- No one can have more than one father
    -- or mother
    all t: Time | all p: Person |
    lone (p.parent.t & Man)
    and
    lone (p.parent.t & Woman)

    ...
```
Revising static constraints

... 

-- A person P's siblings are those people with 
-- the same parents as P (excluding P)

\[
\text{all } t: \text{Time} \mid \text{all } p: \text{Person} \mid \text{some } p.\text{parent}.t \implies p.\text{sibling}.t = \\
\left( \{q: \text{Person} \mid p.\text{parent}.t = q.\text{parent}.t\} - p \right) \text{ else no } p.\text{sibling}.t
\]

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

\[
\text{all } t: \text{Time} \mid \text{all } p: \text{Person} \mid \\
\text{let } p\text{Spouse} = p.\text{marriedWith}.t \mid \\
(p \text{ in Man implies } p\text{Spouse in Woman}) \text{ and } \\
(p \text{ in Woman implies } p\text{Spouse in Man})
\]

...
Revising static constraints

... 

-- A spouse can't be a sibling  
all t: Time | no p: Person | 
  one p.marriedWith.t and 
  p.marriedWith.t in p.sibling.t

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

all t: Time | no p: Person |
let pSpouse = p.marriedWith.t |
  one pSpouse and
  BloodRelatives [p, pSpouse, t]

...
Revising static constraints

... 

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

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

-- the marriedWith relation is symmetric

\[ \text{all } t: \text{Time} \mid \]
\[ \text{marriedWith}.t = \sim(\text{marriedWith}.t) \]
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
  - Add to alive relation
  - NB: No requirement that a person have parents

Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
marrriedWith = {}
children = {}
sibling = {}
alive = {}

Time $t$

Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
marrriedWith = {}
children = {}
sibling = {}
alive = {Sue}

Time $t'$
Transitions

- A person is born to parents
  - Add to alive relation
  - Modify children/parent relations

```
Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
mixedWith = {Matt,Sue, Sue,Matt}
children = {}
sibling = {}
alive = {Matt, Sue}
```

```
Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
mixedWith = {(Matt,Sue), (Sue,Matt)}
children = {(Matt,Sean), (Sue,Sean)}
sibling = {}
alive = {Matt, Sue, Sean}
```
State Sequences

Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
marrriedWith = {}
children = {}
sibling = {}
alive = {Sue}

Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
marrriedWith = {(Matt, Sue), (Sue, Matt)}
children = {}
sibling = {}
alive = {Sue, Matt}

Person = {Matt, Sue, Sean}
Man = {Matt, Sean}
Woman = {Sue}
marrriedWith = {(Matt, Sue), (Sue, Matt)}
children = {(Matt, Sean), (Sue, Sean)}
sibling = {}
alive = {Sue, Matt, Sean}
Express a transition in Alloy

- **Pre condition**
  - Describes the states in which the transition can successfully perform its intended function

- **Post condition**
  - Describes the effects of the transition in generating the next state

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

Distinguishing the pre, post and frame conditions in comment provides useful documentation.
Example: Marriage

```latex
pred marriage [m: Man, w: Woman, t, t': Time, ] { 
  -- precondition
  -- m and w must be alive before marriage
  m+w in alive.t
  -- they must not be married
  no (m+w).marriedWith.t
  -- they must not be blood relatives
  not BloodRelatives [m, w, t]
  -- post-condition
  -- After marriage w is m’s wife
  m.marriedWith.t'= w
  -- After marriage m is w’s wife
  -- (redundant)
  -- frame condition ??
}
```
Frame condition

Which relations are untouched by marriage?

- 5 relations:
  - children, parent, sibling
  - marriedWith
  - alive

- parent and sibling relations are defined in terms of children relation

- Thus, the frame condition has only to consider children, marriedWith and alive relations
Frame condition predicates

pred noChildrenChangeExcept [ps: set Person t, t': Time] {
  all p: Person - ps |
  p.children.t' = p.children.t
}

pred noMarriedWithChangeExcept [ps: set Person t, t': Time] {
  all p: Person - ps |
  p.marriedWith.t' = p.marriedWith.t
}

pred noAliveChange [t, t': Time] {
  alive.t' = alive.t
}
Example: Marriage

pred marriage [m: Man, w: Woman, t,t': Time] {
    -- precondition
    m+w in State.alive.t
    no (m+w).marriedWith.t
    not BloodRelatives [m, w, t]
    -- post-condition
    m.marriedWith.t' = w
    -- frame condition
    noChildrenChangeExcept [none, t, t']
    noMarriedWithChangeExcept [m+w, t, t']
    noAliveChange [t, t']
}
Instance of marriage

open ordering [Time] as T

... 

pred marriageInstance {
  some t: Time |
  some m: Man | some w: Woman |
  let t' = T/next [t] |
  marriage [m, w, t, t']
}

run { marriageInstance }
Example: Birth

pred birth [t, t': Time]
{
  -- precondition and post-condition
  one p: Person |
  p !in alive.t and
  alive.t' = alive.t + p
  -- frame condition
  noChildrenChangeExcept [none, t, t']
  noMarriedWithChangeExcept [none, t, t']
}
Example: Birth from parents

```plaintext
definition birthfromParents [m, w: Person, t, t': Time] {
  -- precondition
  m + w in alive.t
  m.marriedWith.t = w
  -- precondition and post-condition
  one p: Person | {
    -- precondition
    p !in alive.t
    -- postcondition
    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']
  noMarriedWithChangeExcept [none, t, t']
}
```
Instance of birth

pred birthInstance {
    some t: Time |
    let t' = T/next [t] |
    birth [t, t']
}

pred birthFromParentsInstance {
    some t: Time |
    some m, w: Person |
    let t' = T/next [t] |
    birthFromParents [m, w, t, t']
}
Specifying an initial state: 

\begin{verbatim}
pred init [t: Time] {
    no children.t
    no marriedWith.t
    no alive.t
}
\end{verbatim}
Specification of a trace

```plaintext
fact Trace {
  init [T/first]
  all t: Time - T/last |
    let t’ = T/next [t] |
    birth [t, t’] or
    one m: Man | one w: Woman |
    marriage [m, w, t, t’] or
    birthFromParents [m, w, t, t’]
}
run {Trace and some Man and some Woman}
```
Realism constraint

```plaintext
run {  
  marriageInstance  
  birthInstance  
  birthFromParentsInstance  
} for 5
```
Constraint on alive relation

-- only living people can have
-- children or be married

```plaintext
fact staticAlive {
  all t: Time | all p: Person | let mainRels = children + marriedWith |
  p !in alive.t implies ( 
    no p.mainRels.t 
    and 
    no q: Person | p in q.mainRels.t 
  )
}
```
Exercises

- Load family-7.als
- Execute it
- Look at the generated instance
- Does it look correct?
- What if anything would you change about it?