CIS 771: Software Specifications

Lecture -- Week 6: Extended Airport Model

“Airport” Modeling Example

- Enrich the static model of airport resources
  - More realistic physical constraints
  - More realistic ground traffic control policies
- Model movement of mobile resources
For you to do (pause here)

- Read the description of the extended airport problem
- Don’t look ahead at the model (yet)
- Think about how you would modify the model
  - Domains?
  - Sets?
  - Relations?
  - Constraints?

Basic Model

```plaintext
model GroundControl {
  domain {Equipment, Location}

  state {
    partition Plane, Vehicle : Equipment
    partition Runway, Taxiway, Gate : Location
    equipment (~location) : Location! -> Equipment
  }
}
```
CIS 771 -- Extended Airport Model

Main Difference

Fixed resources include gates, runways, and taxi-ways. In this problem we would like to enrich the model of these fixed resources to capture constraints on their topology and inter-connections. Examples constraints (which are detailed below) include: that a gate is comprised of a single location, that runways are comprised of multiple locations, and that runways may intersect but no other fixed resource may. The novelty here is the decoupling of the notion of location from the fixed resources. In the enriched model fixed resources are related to the locations that constitute the resource rather than being thought of as locations themselves.

Fixed Resource Constraints

1. Mobile resources can be at one location at a time
2. A location can hold at most one plane
3. Fixed resources consist of some number of adjacent locations
4. Gates consist of a single location that is adjacent to a single taxiway at a single location
5. Runways consist of multiple locations
6. Of the fixed resources, only runways can intersect and at most one location
7. Taxiways are adjacent to other fixed resources at at most one location
8. Planes can reach a runway from any gate
Fixed Resource Constraints

1. Mobile resources can be at one location at a time
2. A location can hold at most one plane
3. Fixed resources consist of some number of adjacent locations
4. Gates consist of a single location that is adjacent to a single taxiway at a single location
5. Runways consist of multiple locations
6. Of the fixed resources, only runways can intersect and at most one location
7. Taxiways are adjacent to other fixed resources at at most one location
8. Planes can reach a runway from any gate

Enriched Model

model GroundControl {
  domain {MobileResource, FixedResource, Location}
  state {
    partition Plane, Vehicle : MobileResource
    partition Runway, Taxiway, Gate : FixedResource
    location (~equipment) : MobileResource -> Location!
    locations (~partof) : FixedResource -> Location+
    adjacent : Location -> Location
  }
}

CIS 771 -- Extended Airport Model
Location Adjacency

Want to express the basic properties of adjacency of locations

Commutativity  \((l1,l2) \leftrightarrow (l2,l1)\)

Boundedness

Could model a grid with separate relations for each direction

Location adjacency

inv adjacent {
// Adjacency is commutative
    all l1, l2 : Location | l1 in l2.adjacent -> l2 in l1.adjacent
}

### Fixed Resource Constraints

1. Mobile resources can be at one location at a time
2. A location can hold at most one plane
3. Fixed resources consist of some number of adjacent locations
4. Gates consist of a single location that is adjacent to a single taxiway at a single location
5. Runways consist of multiple locations
6. Of the fixed resources, only runways can intersect and at most one location
7. Taxiways are adjacent to other fixed resources at at most one location
8. Planes can reach a runway from any gate

### Fixed resource constraints

```plaintext
inv FixedResourceConstraint2 { all l : Location | sole (l.equipment & Plane) }

inv FixedResourceConstraint4 { all g : Gate | one g.locations all g : Gate | one (g.locations.adjacent.partof & Taxiway) && one g.locations.adjacent }
```
Fixed Resource Constraints

1. Mobile resources can be at one location at a time
2. A location can hold at most one plane
3. Fixed resources consist of some number of adjacent locations
4. Gates consist of a single location that is adjacent to a single taxiway at a single location
5. Runways consist of multiple locations
6. Of the fixed resources, only runways can intersect and at most one location
7. Taxiways are adjacent to other fixed resources at most one location
8. Planes can reach a runway from any gate

Fixed resource constraints

inv FixedResourceConstraint5 { 
    all f : Runway | not sole f.locations 
}

inv FixedResourceConstraint6 { 
    all nr : FixedResource - Runway | 
        all f : FixedResource | 
            nr != f -> no (nr.locations & f.locations) 
    all r1, r2 : Runway | 
        r1 != r2 -> sole (r1.locations & r2.locations) 
}
**Fixed Resource Constraints**

1. Mobile resources can be at one location at a time
2. A location can hold at most one plane
3. Fixed resources consist of some number of adjacent locations
4. Gates consist of a single location that is adjacent to a single taxiway at a single location
5. Runways consist of multiple locations
6. Of the fixed resources, only runways can intersect and at most one location
7. Taxiways are adjacent to other fixed resources at at most one location
8. Planes can reach a runway from any gate

---

**Fixed resource constraints**

```plaintext
inv FixedResourceConstraint7 {
    all t : Taxiway | all f : FixedResource |
    t != f -> no (t.locations & f.locations) &&
    sole (t.locations.adjacent &
         f.locations.adjacent)
}
inv FixedResourceConstraint8 {
    all g : Gate |
    some (g.locations.+adjacent & Runway.locations)
}
```
For you to do (pause here)

- Load airport-5.all
- Instantiate the model and look at the results
- Enforce some realism constraints
  - Some gate, planes, service vehicles
  - Some intersecting runways

Mobile Resource Constraints

9. Mobile resources can only move between adjacent locations
10. Only planes can be on runways.
11. At most one plane can be on a runway.
12. Gates can have a single plane.
13. Gates can have multiple service vehicles.
14. If a plane is at a gate, then a vehicle should be there to service it.
15. A plane can take off on a runway when all intersecting runways are empty.
16. A plane can land on a runway when it is empty
Mobile Resource Constraints

9. Mobile resources can only move between adjacent locations.
10. Only planes can be on runways.
11. At most one plane can be on a runway.
12. Gates can have a single plane.
13. Gates can have multiple service vehicles.
14. If a plane is at a gate, then a vehicle should be there to service it.
15. A plane can take off on a runway when all intersecting runways are empty.
16. A plane can land on a runway when it is empty.

Fixed resource constraints

inv MobileResourceConstraint10 {
    all r : Runway | r.locations.equipment in Plane
}

inv MobileResourceConstraint11 {
    all r : Runway | sole r.locations.equipment
}
Mobile Resource Constraints

9. Mobile resources can only move between adjacent locations
10. Only planes can be on runways.
11. At most one plane can be on a runway.
12. Gates can have a single plane.
13. Gates can have multiple service vehicles.
14. If a plane is at a gate, then a vehicle should be there to service it.
15. A plane can take off on a runway when all intersecting runways are empty.
16. A plane can land on a runway when it is empty.

Fixed resource constraints

```
inv MobileResourceConstraint12-13 {
    all g : Gate | sole (g.locations.equipment & Plane)
}
inv MobileResourceConstraint14 {
    all g : Gate |
        some (g.locations.equipment & Plane) ->
        some (g.locations.equipment - Plane)
}
```
For you to do (pause here)

- Load airport-6.all
- Instantiate the model and look at the results
- Enforce some realism constraints
  - Some gate, planes, service vehicles
  - Some intersecting runways
- Pose some queries about the model
  - Is there a runway that cannot be reached from some gate?

Movement Constraints

9. Mobile resources can only move between adjacent locations
10. Only planes can be on runways.
11. At most one plane can be on a runway.
12. Gates can have a single plane.
13. Gates can have multiple service vehicles.
14. If a plane is at a gate, then a vehicle should be there to service it.
15. A plane can take off on a runway when all intersecting runways are empty.
16. A plane can land on a runway when it is empty
## Airplane Takeoff

Explicit constraints
- on a runway
- when all intersecting runways are clear
- results in the plane being airborne

Other constraints
- no other vehicles move

## Airplane Landing

Explicit constraints
- not on a runway (or taxiway or at gate)
- when runway is clear
- results in the plane being on runway

Other constraints
- no other vehicles move
What about Fixed Resources?

- Can new ones be added?
- Can the locations comprising a resource change?
- Can the topology of FixedResources at the airport change?
- Can the topology of locations within a FixedResource change?

Mutabilities

domain {MobileResource, fixed FixedResource, fixed Location}
state {
    partition Runway, Taxiway, Gate : fixed FixedResource
    partition Plane, ServiceVehicle : static MobileResource
    location (~equipment) : MobileResource -> Location!
    locations (~partof) : static FixedResource -> static Location+
    adjacent : static Location -> static Location
}

**Operation Methodology**

Identify pre, post, and frame-conditions

Define global frame-conditions using mutabilities
  - combine fixed domains/sets with static relations to achieve fixed relations

Define parameterized condition for pre-condition
  - instantiate condition as sanity check of operation
  - reference condition inside operation to eliminate redundancy

Define post-condition and operation specific frame-condition explicitly referencing primed components

---

**Takeoff Pre-condition**

```plaintext
cond TakeoffPrecondition(p : Plane!, r : Runway!) {
    p.location in r.locations
    all ir : Runway |
        (one (ir.locations & r.locations)) ->
        no ir.locations.equipment
}
```
**Takeoff Operation**

```plaintext
op Takeoff(p : Plane!, r : Runway!) {
    TakeoffPrecondition(p,r)
    // Post conditions
    p.location' !in (Runway+Gate+Taxiway).locations
    no r.locations.equipment'
    // Frame conditions
    all rw : Runway - r |
        rw.locations.equipment' = rw.locations.equipment'
}
```

**Landing Pre-condition**

```plaintext
cond LandPrecondition(p : Plane!, r : Runway!) {
    p.location !in (Runway + Gate + Taxiway).locations
    no r.locations.equipment
}
```
Landing Operation

```plaintext
op Land(p : Plane!, r : Runway!) {
    LandPrecondition(p,r)
    // Post conditions
    p.location' in r.locations
    r.locations.equipment' = r.locations.equipment + p
    // Frame conditions
    all rw : Runway - r |
        rw.locations.equipment' = rw.locations.equipment
}
```

For you to do (pause here)

Load airport-7.all
Instantiate the operator preconditions
- Can you produce an instance?
- If not, perhaps you need adjust the scopes.
Instantiate the operators
- Do the post-states look right?
- Are the changes enforced by the post-condition apparent?
- Are there unexpected changes?
Check operations against selected invariants
For you to do (pause here)

Other operations to think about
- New plane landing on a runway
- Plane taxiing
- Service vehicle moving
- Pulling into gate
- Backing out of gate

Other Vehicle Movement
- Constraint 9 requires movement to occur between adjacent locations
- Connectivity of FixedResources is therefore a property of adjacency of locations
- Can a plane taxi from any gate to some runway?
- Can a plane taxi from any gate to any runway? (stronger)
Necessary Condition for Taxiing

assert GateRunway {
    all gate : Gate |
    all runway : Runway |
    runway in gate.locations.+adjacent.partof
}

For you to do

Load airport-7.all
Check the assertion for the reachability of runways from gates?
Visualize the counter-example
  - Using ACAs facilities
  - or on your own

How could you constrain the system to enforce reachability of a runway from every gate?