Boarding Pass Issuance to Passengers at Airport

By

Soe Zarni
Bargava Subramanian
University of Maryland
December 6, 2005
Airport authorities have fixed (constrained) space in airport which the airlines can use to set up counters to issue boarding pass.

Passengers of various airlines come and wait in the space in the form of queue to obtain their boarding pass.

**Requirements:**

**Airlines**: Issue boarding pass to passengers

**Passenger**: obtain boarding pass
A specific example:

Fixed room area where three counters are set up.
Two airlines operate.
Each owns one counter
They share the third counter.

<table>
<thead>
<tr>
<th>Airline A</th>
<th>Counter 1</th>
<th>Counter 2 (Share)</th>
<th>Airline B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>varies</td>
<td>Counter 3</td>
</tr>
</tbody>
</table>
Our two-fold approach

Solving Spatial Constraint
• How do the airlines determine their operating space?
• How should their queue structure be?

Solving Temporal Constraint
• Do the passengers catch the flight on time? - we do not consider
• Do the airlines process the passengers so that they catch the flight?
  - Yes – but there is an assumption – to avoid complications – we assume that if airline obtain extra counter – they process the extra burden and are well-off.
Accomplish the following tasks:

• Determine number of operating counters required at a specific time
• Given operating space for the airlines, the queue length has to be determined.

Both of the above problems are inherently the same.

Procedure:

1) Start each airline with just one counter and with just one queue line
2) Check periodically the population of the queue.
3) Whenever the queue becomes congested, add one more line
4) When the whole counter space is nearing capacity, request for the next counter.
5) Temporal logic provides solution of when next counter is made available.
6) Reassign population amongst both the counters in a fair manner.
7) Release counter when done or when requested.
Binary Space Partitioning Tree

• Binary space partitioning (BSP) is a method for recursively subdividing a space into convex sets by hyperplanes.

• This subdivision gives rise to a representation of the scene by means of a tree data structure known as a BSP tree.

"BSP tree construction is a process which takes a subspace and partitions it by any hyperplane that intersects the interior of that subspace. The result is two new subspaces that can be further partitioned by recursive application of the method."

• Our problem is a BSP tree data-structure - variant.

• A point is represented by a vector P(x,y,z)

• A plane is represented by a vector (P,p). P is the normal vector to plane.

Possible Extensions: Incorporate Quad-trees for queue formation. Quad-trees recursively divide given subspace into 4 subspaces.
• Temporal logic is validated using UPPAAL.

“Uppaal is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata”

• Model Each automaton

• For each automaton, specify entry and exit conditions.

• Two exit conditions – Guard and Invariant

• Guard – exit condition based on an event happening at/within a particular time

• Invariant – upperbound for the state – max.(or min.) time one spends on the state.

• Synchronization – Trigger an event in another automaton on satisfaction of some constraint of this automaton.

• Templates(or Events) communicate through edges – having constraints.
• State Check:

(i) Reachability – Can we reach from State A to State B?
(ii) Safety – Can two events occur simultaneously?
(iii) Liveness – if some event happens – does it trigger another event at some later point of time?

• UPPAAL has 3 screens:

(i) System editor – where all timed automata and constraints are specified
(ii) Simulator – Active transactions and the Message Sequence Chart is displayed – we can trace flow of any transaction
(iii) Verifier – specify state check properties and see if it is satisfied.
“It took a lot of time to learn the tool – no simple walk-through example provided. Modeling time constraints is a bit tricky. We took quite a lot of time to understand the tool.”

The UPPAAL system we modeled has the following features/constraints

(i) The request for counter happens 30 min before actual counter taking over

(ii) The other airline has 25 min to respond

(iii) If no response for 25 min – the airline which made the request gets the counter

(iv) Setting up of counter takes 5 min

(v) Airline can have the counter for max of 90 min – after that it becomes empty – whoever requests it next gets it.

(vi) Two airlines cannot simultaneously hold the counter

(vii) More verifying properties – on the model – DEMO TIME !
**Cost Evaluation**:

We are currently writing simple optimization equations for the problem. We are considering only the airline perspective.

*To optimize:*

**Minimize Expense**

Where Expense is composed of:

- **Operating Expense**: No. of hours they use the extra counter – composed of materials and personnel used for the extra counter.
- **Passenger Reimbursement**: Amount they need to pay the passenger when he misses the flight.

*Subject to constraints:*

- Passenger Arrival Rate
- Average Process time
- Average counter setting-up time
Spatial Logic Model Checker:

We are using the SLMC tool that allows the user to automatically verify behavioral and spatial properties of distributed concurrent systems to validate spatial logic for a simplified case.


More about Cost

UPPAAL CORA is a branch of UPPAAL for Cost Optimal Reachability Analysis. Uses the timed automata LPTA.
Thank You Dr. Austin

Thank You!

Questions?