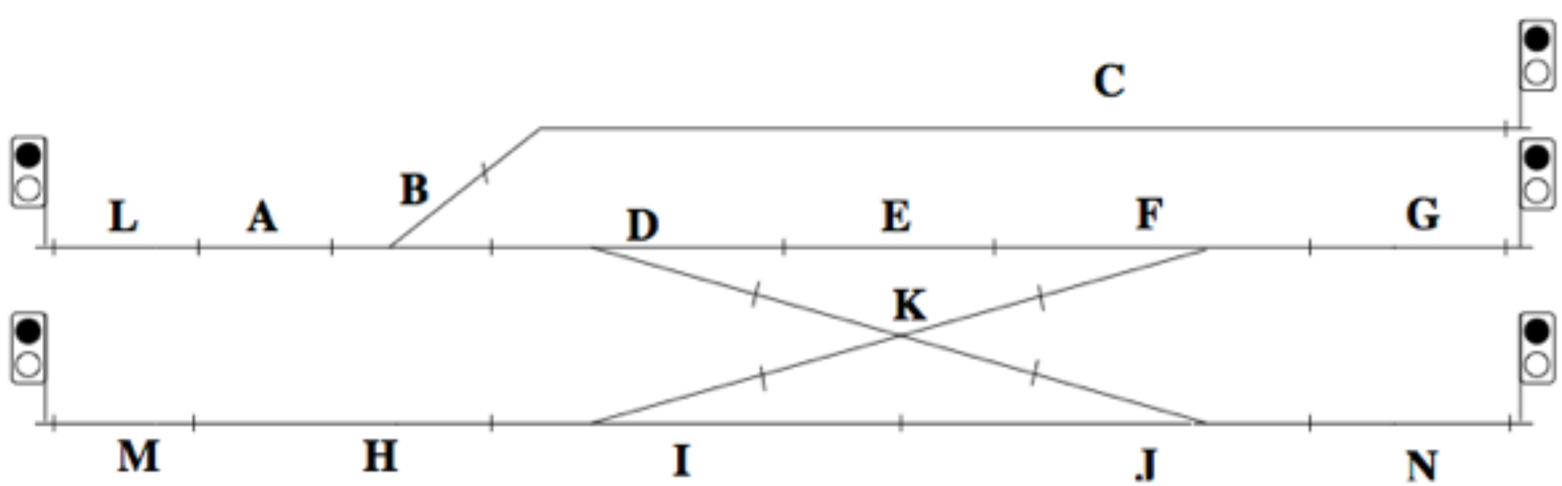
UN LOCKING the Mysteries of an INTER LOCKING Model

Michael Leuschel
Jens Bendisposto, Dominik Hansen
University of Düsseldorf

Advance

Chapter 17

- Formal model of an interlocking system
- Interlocking: safely operate signals and points within an area of the train network
  - no collisions, do not move points while trains drive over, trains reach destination, …

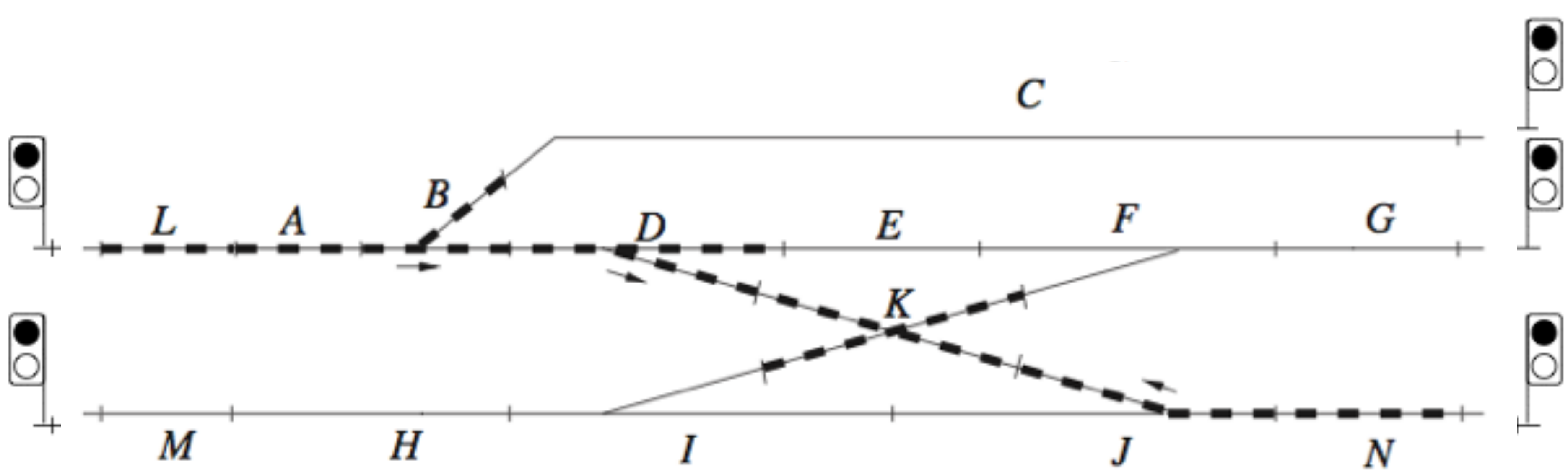


Source: Images from Abrial, Event-B Book

Essentials of Model

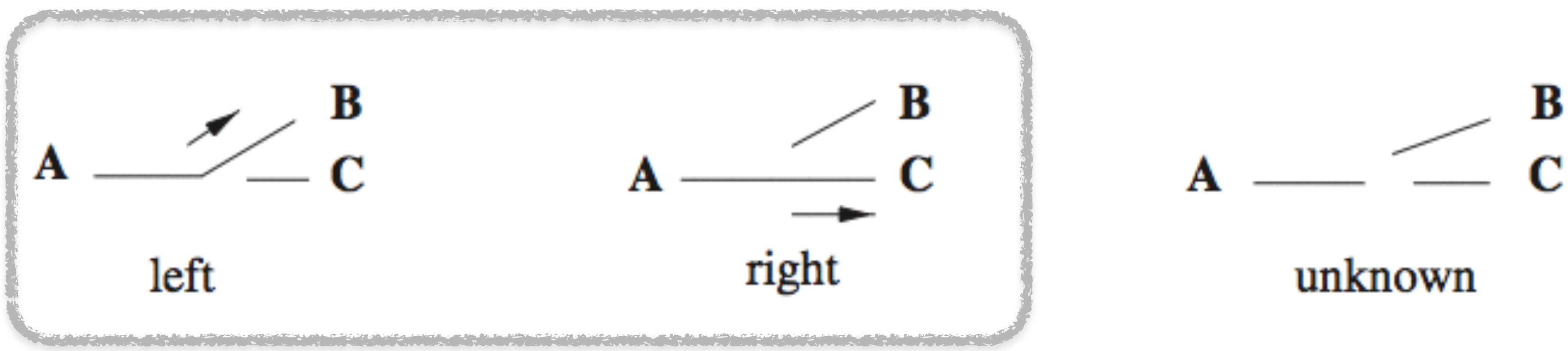
- Network divided into blocks (A-N below)
- Two special components: points (B,D,F,I,J below) and crossings (K below)
- Statically determined routes (R1: L,A,B,D,E,F,G … R10) protected by a signal

R3:



Chapter 17: Comments

- Simplifying assumptions in Chapter 17 model:
  - points only left or right (instantaneous moves),…



- But still “close” to real models
- Validation of big interlockings is challenging (cf. Paris RER renovation project)

Proof, Animation, MC

- Event-B Model fully proven
  ⇒ Why do we need to animate ?

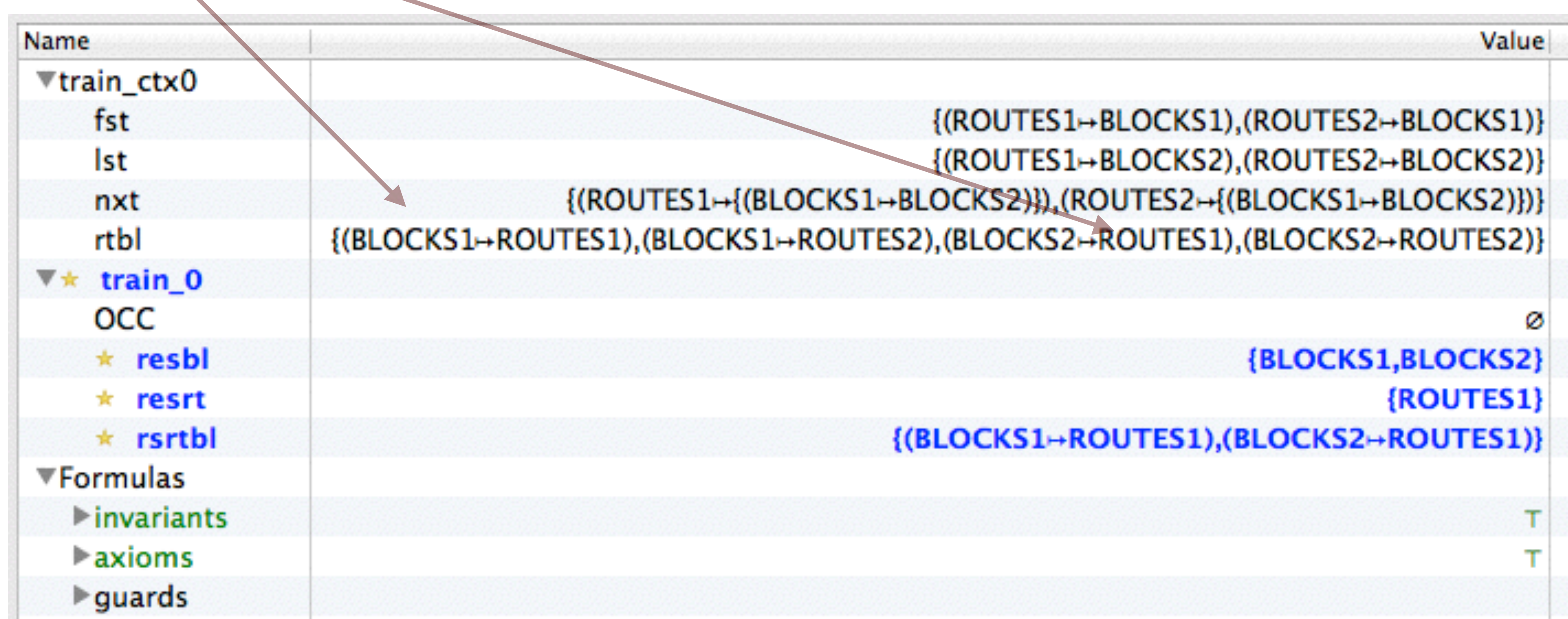
  Why do we need to model check ?

  - Have we proven the right things ?
  - Is the model too restrictive ? (Deadlocks)
  - Is the model too permissive (undesirable behaviour, configurations) ?

Animation (with ProB)

- Without providing topology; possible but not so interesting

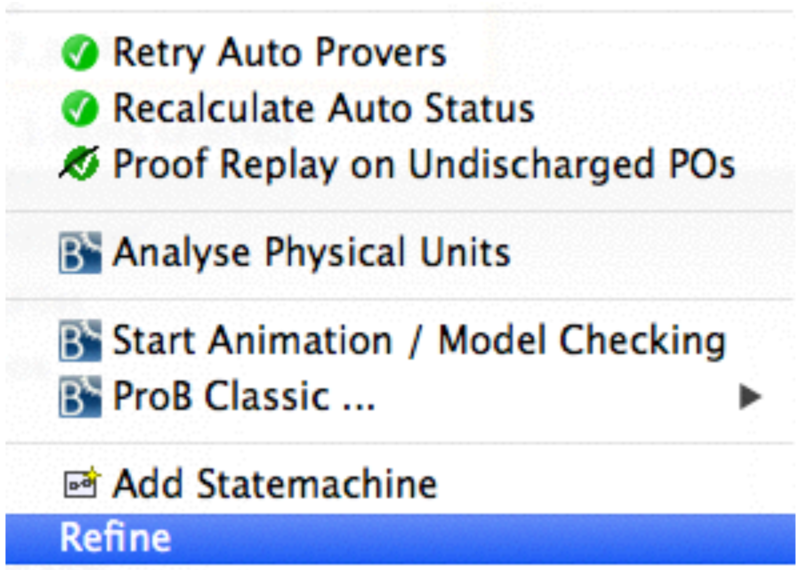
Note: identical routes (not prohibited by axioms)



How to instantiate a model

- Simply add axioms to your context
  Downside: interferes with proof activities

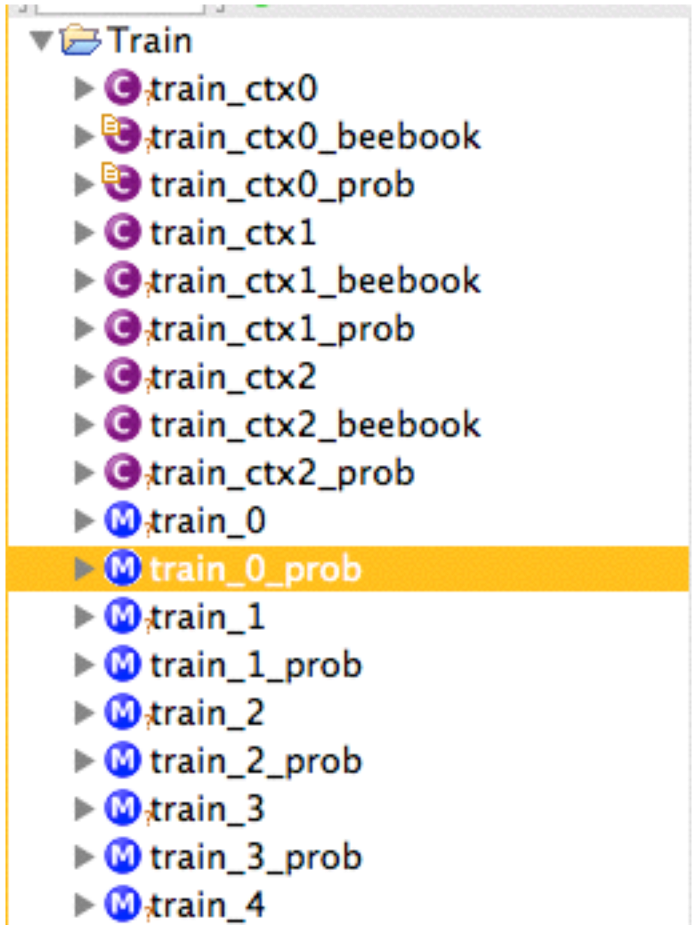
- Better: extend your context, refine your machine to include context

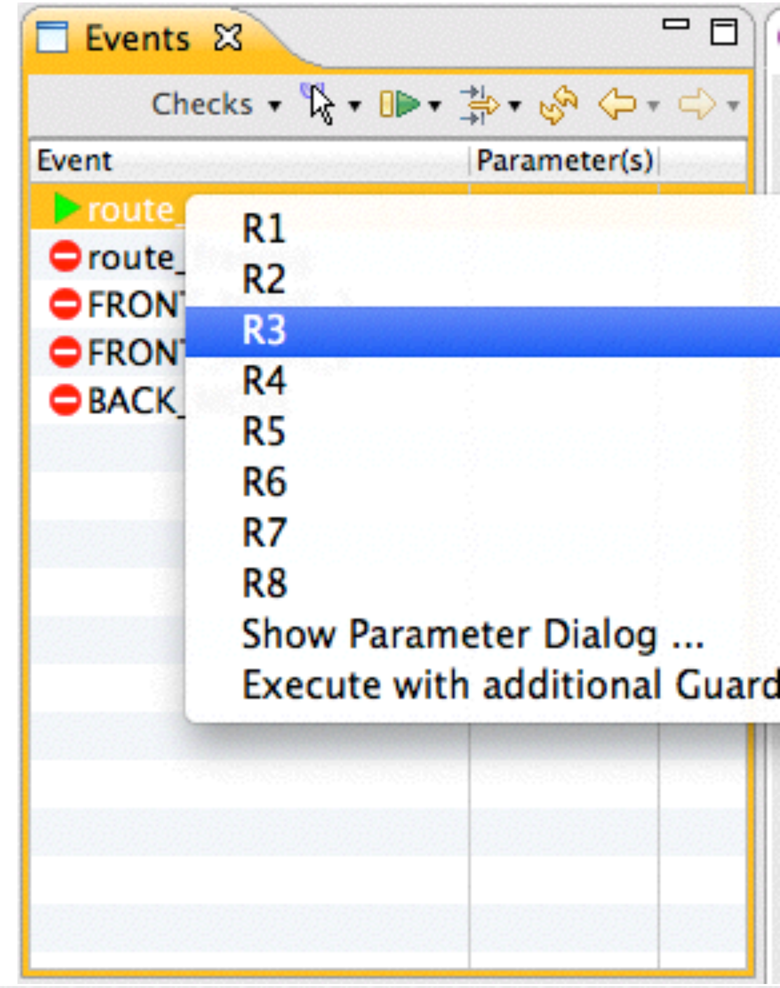


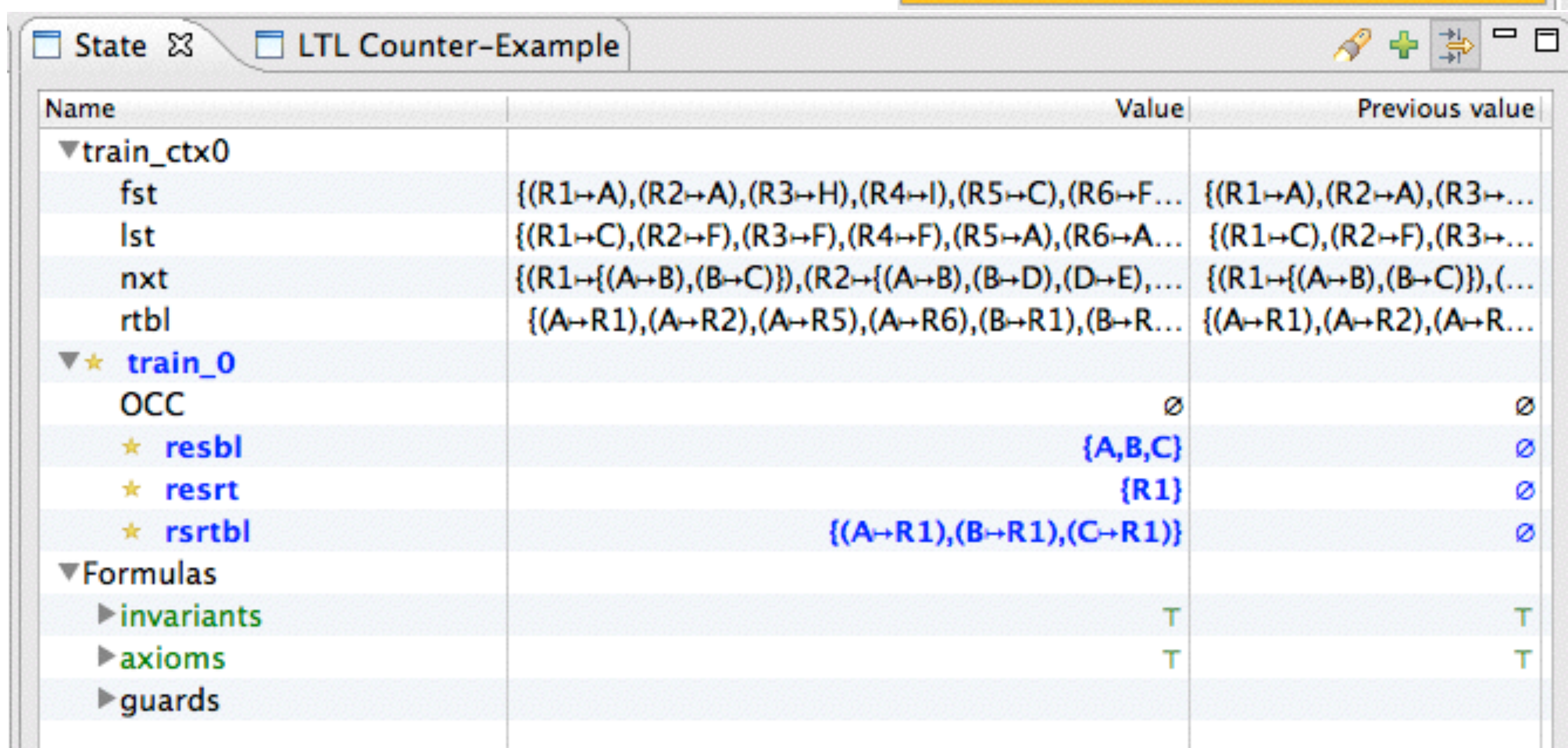

```
context train_ctx0_beebook // contains data for the sample track layout in the Bee Book by
 extends train_ctx0
constants A B C D E F G H I J K L M N R1 R2 R3 R4 R5 R6 R7 R8 R9 R10

axioms
  @axm44 partition(BLOCKS, {A}, {B}, {C}, {D}, {E}, {F}, {G}, {H}, {I},{J}, {K},{L},{M},{N}
  @axm45 partition(ROUTES, {R1}, {R2}, {R3},{R4},{R5},{R6},{R7},{R8},{R9},{R10})


  @compute_rtbl_from_nxt rtbl = {b↦r| r∈dom(nxt) ∧ (b∈dom(nxt(r)) ∨ b∈ran(nxt(r)))}
  @axm40 nxt = {(R1 ↦ {L↦A, A↦B,B↦C}),
    (R2 ↦ {L↦A, A↦B,B↦D,D↦E,E↦F, F↦G}),
    (R3 ↦ {L↦A, A↦B,B↦D,D↦K,K↦J, J↦N}),
    (R4 ↦ {M↦H,H↦I,I↦K,K↦F,F↦G}),
    (R5 ↦ {M↦H,H↦I,I↦J,J↦N}),
    (R6 ↦ {C↦B,B↦A,A↦L}),
    (R7 ↦ {G↦F,F↦E,E↦D,D↦B,B↦A,A↦L}),
    (R8 ↦ {N↦J,J↦K,K↦D,D↦B,B↦A,A↦L}),
    (R9 ↦ {G↦F,F↦K,K↦I,I↦H,H↦M}),
    (R10 ↦ {N↦J,J↦I,I↦H,H↦M})}
  @axm41 fst = {(R1 ↦ L),(R2 ↦ L),(R3 ↦ L),
                (R4 ↦ M),(R5 ↦ M),
                (R6 ↦ C),
                (R7 ↦ G),(R8 ↦ N),
                (R9 ↦ G),(R10 ↦ N)}
```

```
machine train_0_prob refines train_0
sees train_ctx0_prob
…
end
```

Rodin feature request: extend all events

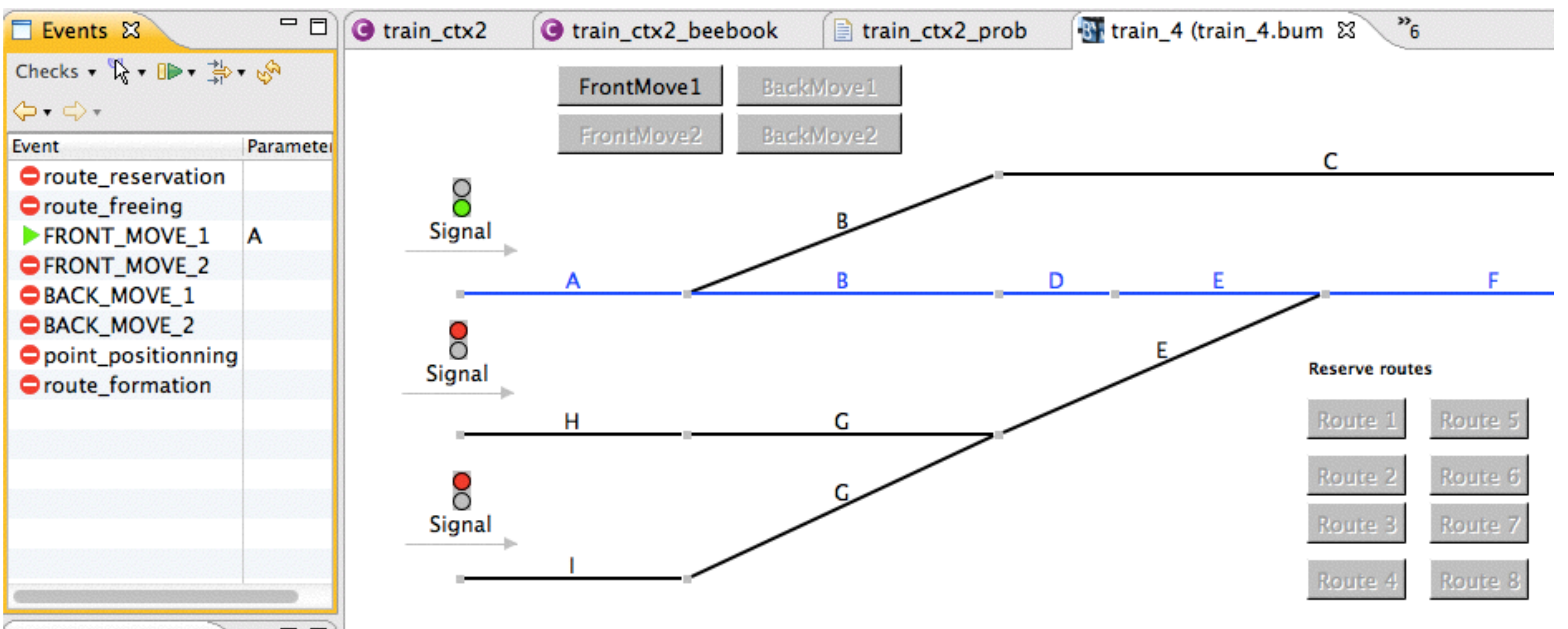

Train
train_ctx0
train_ctx0_beebook
train_ctx0_prob
train_ctx1
train_ctx1_beebook
train_ctx1_prob
train_ctx2
train_ctx2_beebook
train_ctx2_prob
train_0
train_0_prob
train_1
train_1_prob
train_2
train_2_prob
train_3
train_3_prob
train_4


Events
Checks
Event
Parameter(s)
route_
route_
FRONT
FRONT
BACK_
R1
R2
R3
R4
R5
R6
R7
R8
Show Parameter Dialog ...
Execute with additional Guard


State
LTL Counter-Example
Name
Value
Previous value
train_ctx0
fst
{(R1↦A),(R2↦A),(R3↦H),(R4↦I),(R5↦C),(R6↦F…
{(R1↦A),(R2↦A),(R3↦…
lst
{(R1↦C),(R2↦F),(R3↦F),(R4↦F),(R5↦A),(R6↦A…
{(R1↦C),(R2↦F),(R3↦…
nxt
{(R1↦{(A↦B),(B↦C)}),(R2↦{(A↦B),(B↦D),(D↦E),…
{(R1↦{(A↦B),(B↦C)}),(…
rtbl
{(A↦R1),(A↦R2),(A↦R5),(A↦R6),(B↦R1),(B↦R…
{(A↦R1),(A↦R2),(A↦R…
train_0
OCC
∅
∅
resbl
{A,B,C}
∅
resrt
{R1}
∅
rsrtbl
{(A↦R1),(B↦R1),(C↦R1)}
∅
Formulas
invariants
T
T
axioms
T
T
guards

Graphical Visualization

Demo: ProB + BMotionStudio



Using the constraint solver

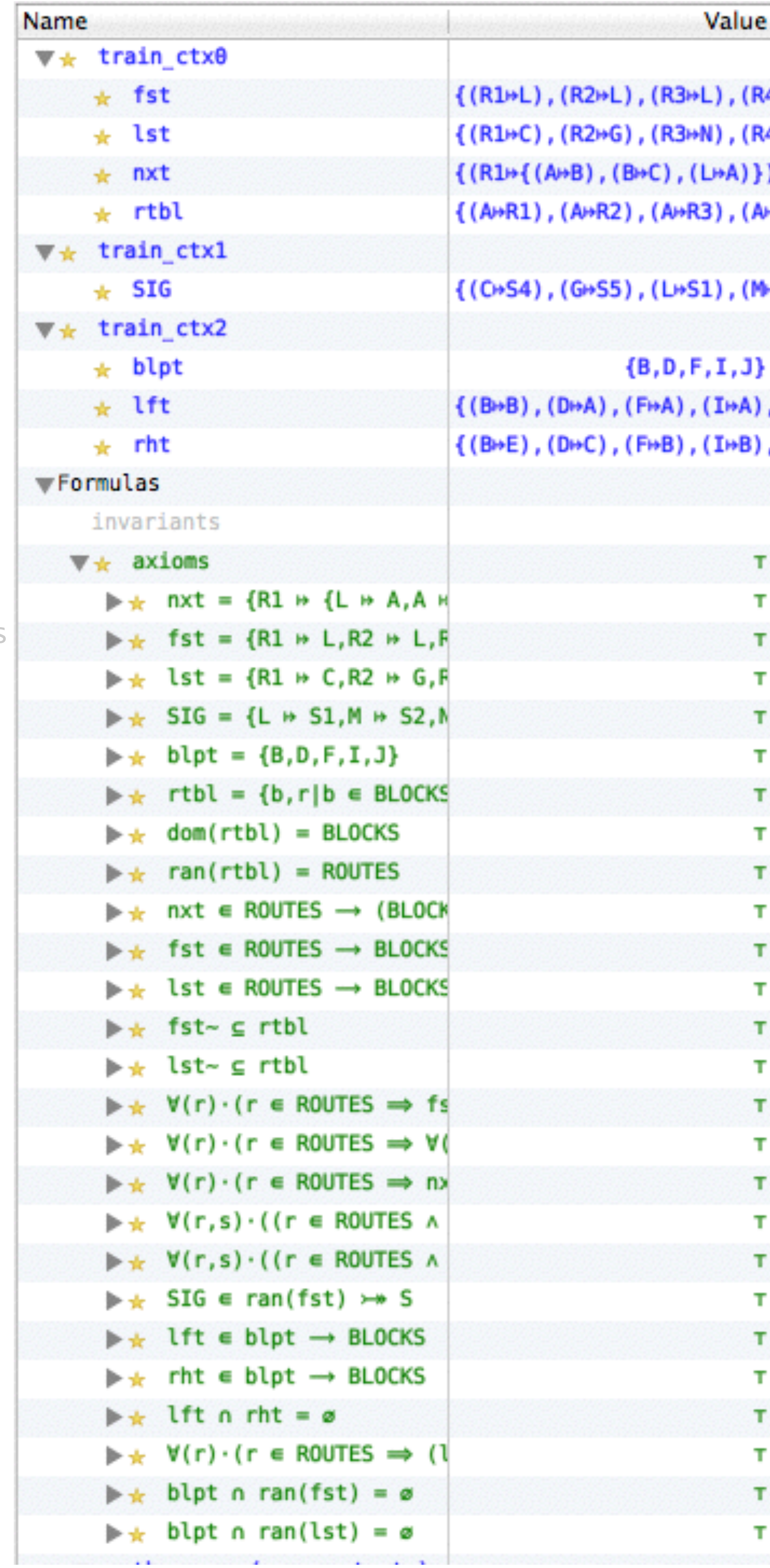
- We do not need to specify all values; we can provide some values and let the ProB constraint solver instantiate the other constants

```
context train_ctx2 extends train_ctx1

constants blpt lft rht

axioms
   @axm1 blpt ⊆ BLOCKS  // blocks with points: sets of blocks containing points
   @axm2 lft ∈ blpt → BLOCKS
   @axm3 rht ∈ blpt → BLOCKS
   @axm4 lft ∩ rht = ∅
   @axm5 ∀r·r∈ROUTES ⇒ (lft ∪ rht) ∩ (nxt(r) ∪ (nxt(r))∼) ∈ blpt↣BLOCKS
   @axm6 blpt ∩ ran(fst) = ∅
   @axm7 blpt ∩ ran(lst) = ∅

end
```

```
context train_ctx2_beebook extends train_ctx2 train_ctx1_beebook
axioms
   @prob_axm1 blpt = {B,D,F,I,J}
end
```

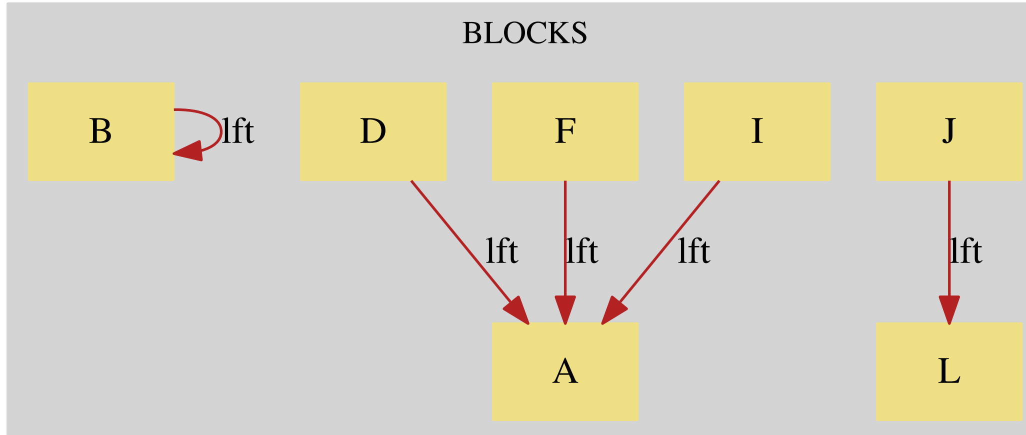
```
context train_ctx2 extends train_ctx1

constants blpt lft rht

axioms
   @axm1 blpt ⊆ BLOCKS  // blocks with points: sets of blocks containing points
   @axm2 lft ∈ blpt → BLOCKS
   @axm3 rht ∈ blpt → BLOCKS
   @axm4 lft ∩ rht = ∅
   @axm5 ∀r·r∈ROUTES ⇒ (lft ∪ rht) ∩ (nxt(r) ∪ (nxt(r))∼) ∈ blpt⤔BLOCKS

   @axm6 blpt ∩ ran(fst) = ∅
   @axm7 blpt ∩ ran(lst) = ∅

end
```

```
context train_ctx2_beebook extends train_ctx2 train_ctx1_beebook
axioms
   @prob_axm1 blpt = {B,D,F,I,J}
end
```



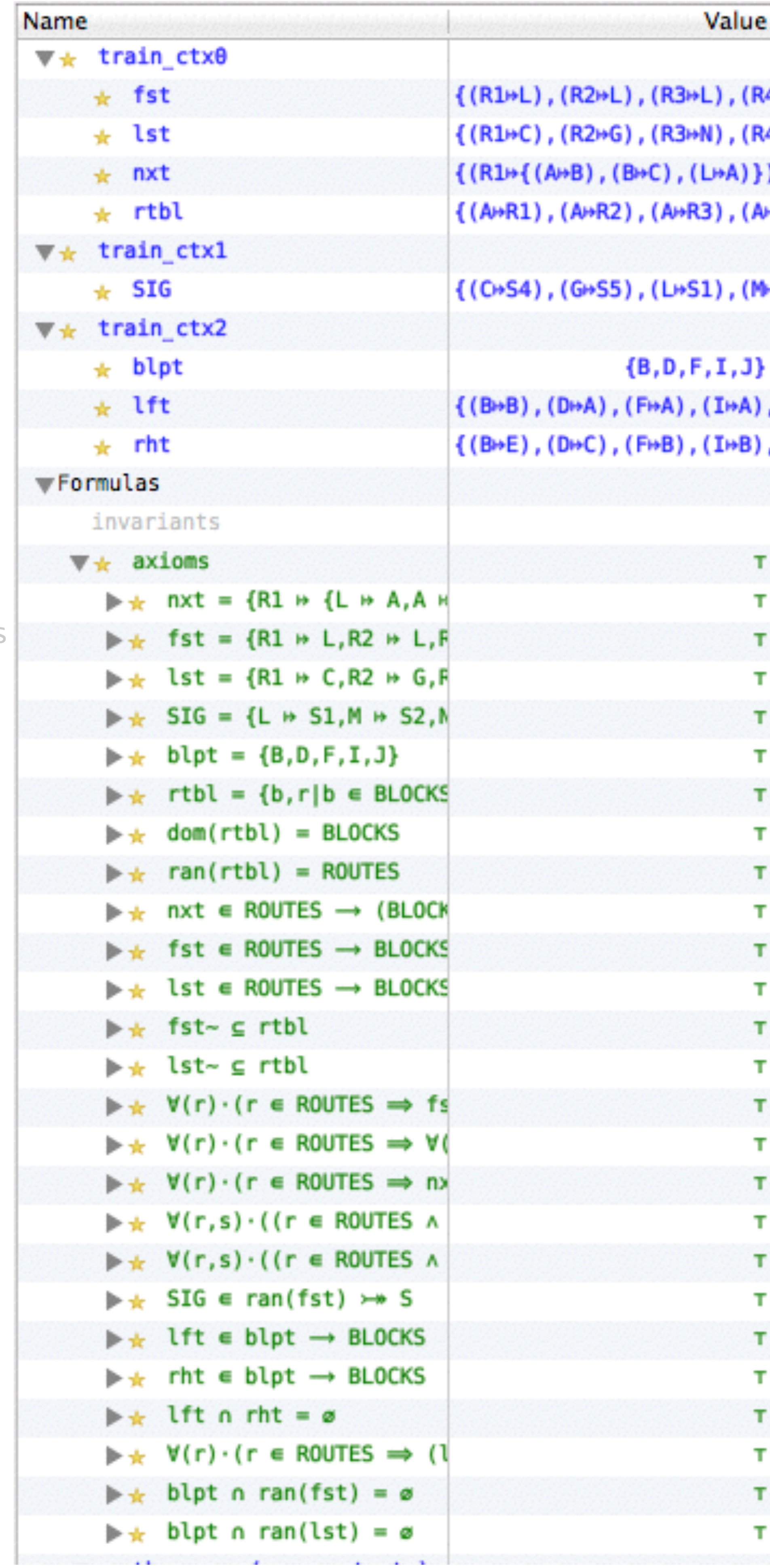
blpt: blocks with point

lft, rht: possible successors of points




```
context train_ctx2 extends train_ctx1

constants blpt lft rht

axioms
   @axm1 blpt ⊆ BLOCKS  // blocks with points: sets of blocks containing points
   @axm2 lft ∈ blpt → BLOCKS
   @axm3 rht ∈ blpt → BLOCKS
   @axm4 lft ∩ rht = ∅
   @axm5 ∀r·r∈ROUTES ⇒ (lft ∪ rht) ∩ (nxt(r) ∪ (nxt(r))~) ∈ blpt⇸BLOCKS
   @axm6 blpt ∩ ran(fst) = ∅
   @axm7 blpt ∩ ran(lst) = ∅

end
```

```
context train_ctx2_beebook extends train_ctx2 train_ctx1_beebook
axioms
   @prob_axm1 blpt = {B,D,F,I,J}
end
```



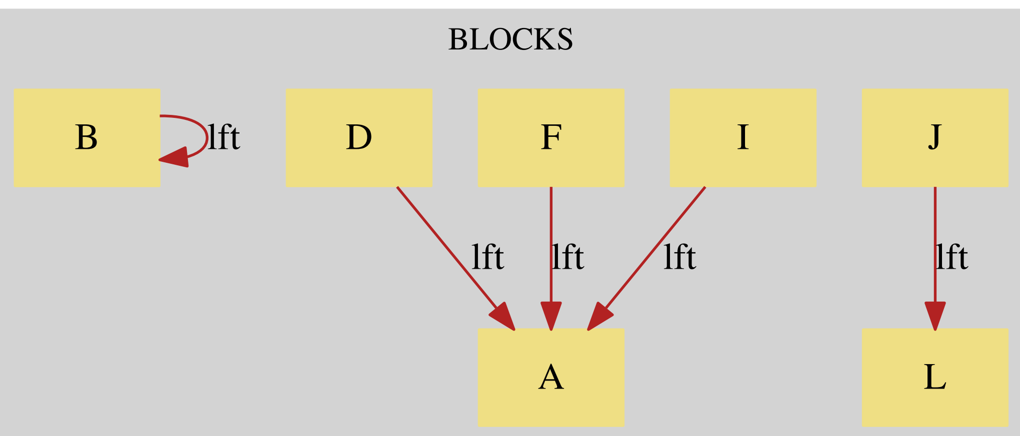
blpt: blocks with point

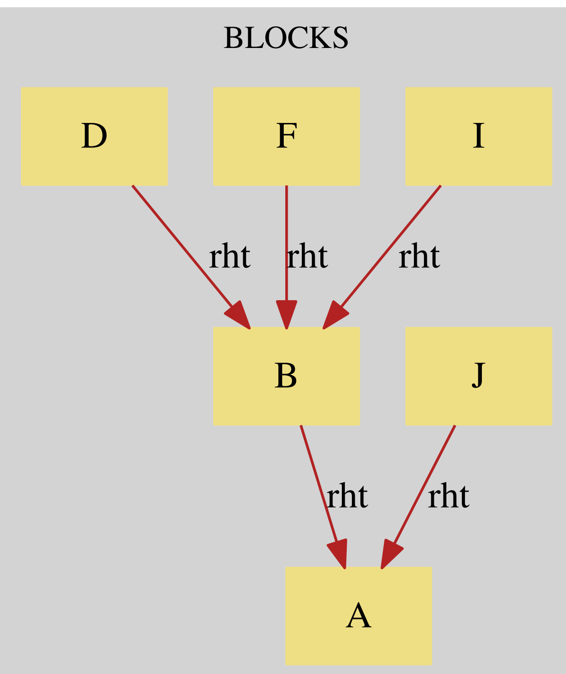
lft, rht: possible successors of points

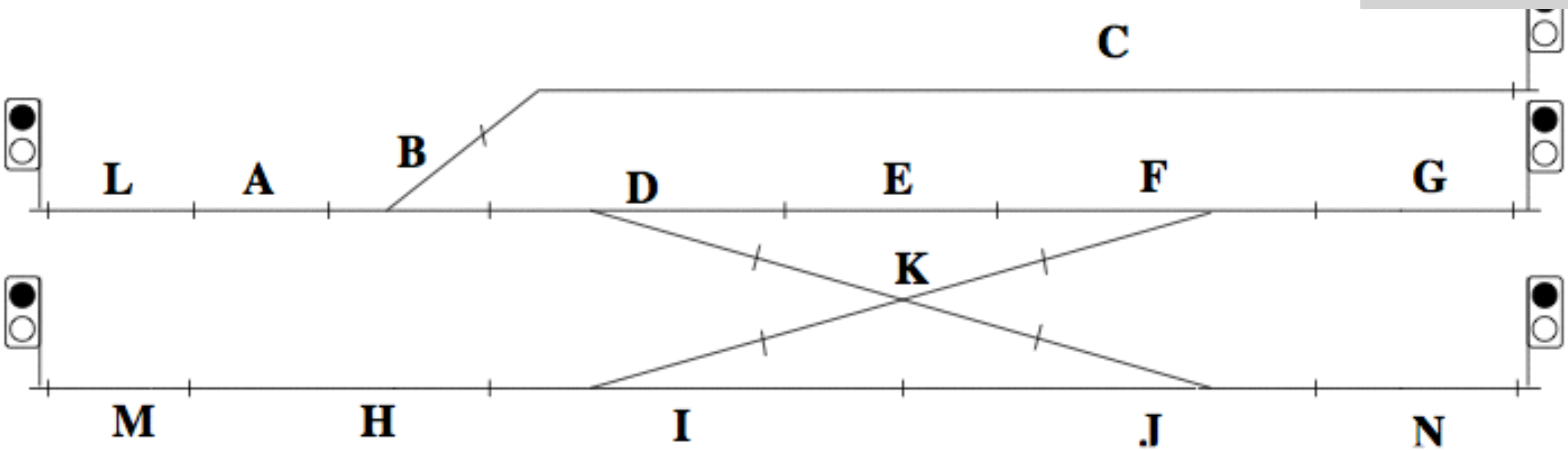
```
@axm2 lft ∈ blpt → BLOCKS
@axm3 rht ∈ blpt → BLOCKS

@axm4 lft ∩ rht = ∅
@axm5 ∀r·r∈ROUTES ⇒ (lft ∪ rht) ∩ (nxt(r) ∪ (nxt(r))∼) ∈ blpt⤔BLOCKS
```

Satisfied by empty set

(choose lft/rht such that empty intersection with nxt,nxt~)





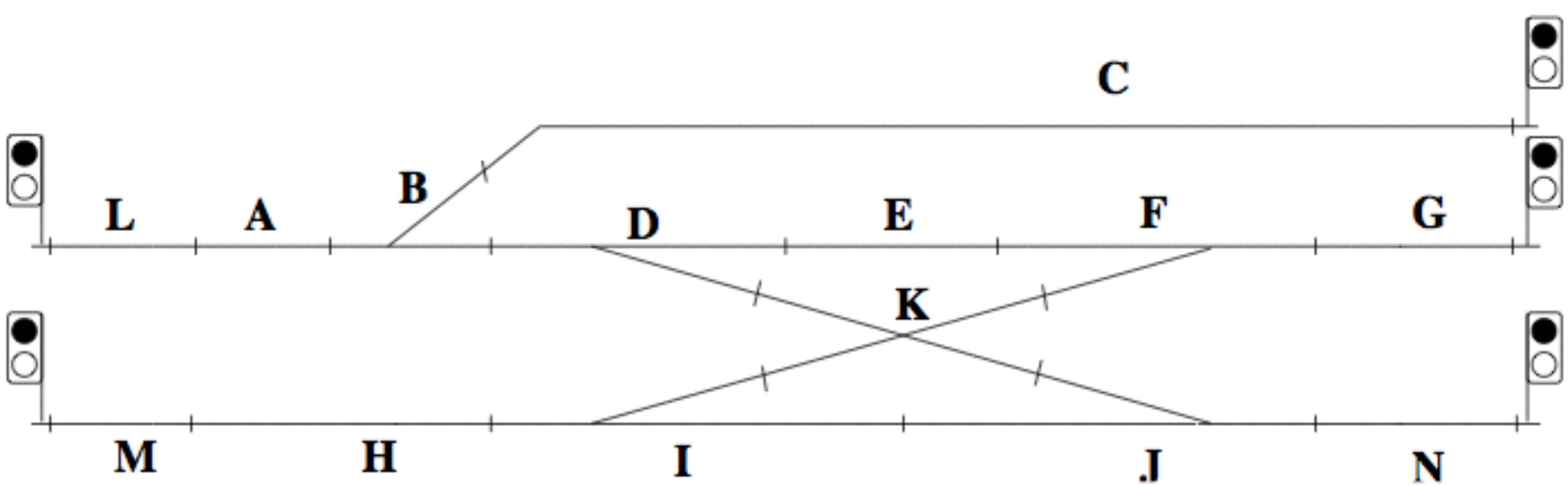


First Conclusion

- Model finding/constraint solving + animation useful in uncovering missing axioms
  - axioms were not required in first four levels of refinement
  - but would have been needed at next levels (book stopped here; no event used lft, rht to move points !)

Model Checking

- The simple topology from the book certainly cannot pose any problem, can it ?
- How many states are there for the first refinement ?



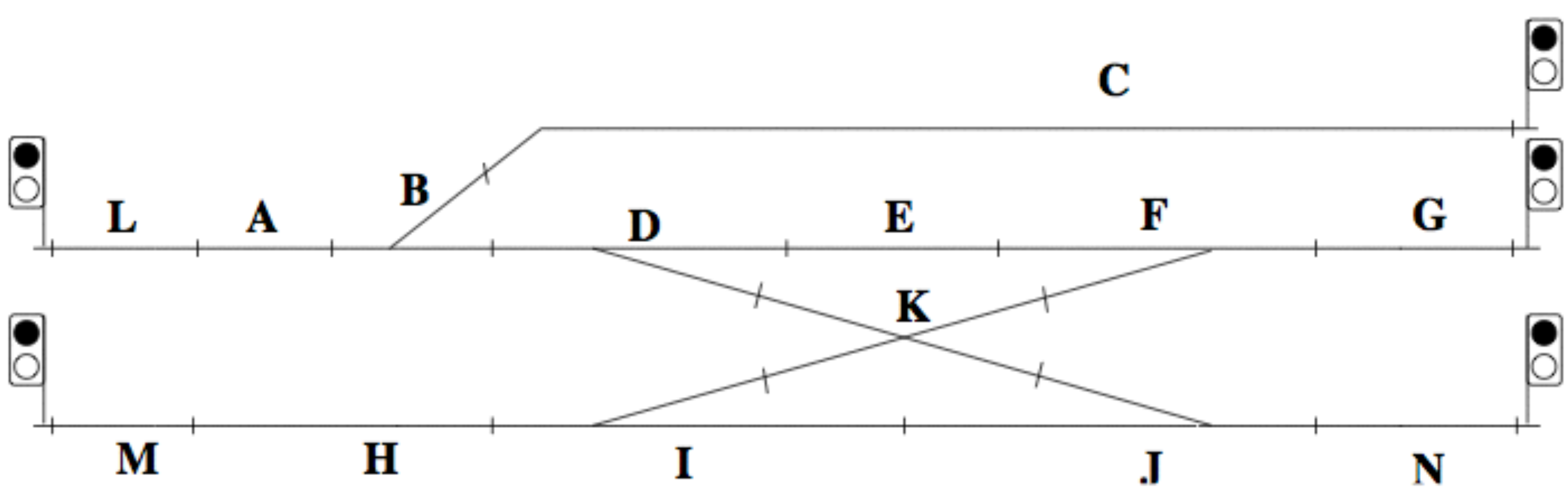
```
TLA+ module '/home/hansen/./Train_1_beebook_v2.tla' created.
Configuration file '/home/hansen/./Train_1_beebook_v2.cfg' created.
...
--------------------------------
TLC2 Version 2.05 of 17 April 2013
Running in Model-Checking mode.
...
Starting... (2014-02-05 11:49:29)
Computing initial states...
Finished computing initial states: 1 distinct state generated.
Progress(5) at 2014-02-05 11:49:36: 152 states generated (152 s/min),
    86 distinct states found (86 ds/min), 38 states left on queue.
Progress(19) at 2014-02-05 11:50:36: 17499 states generated (17347 s/min),
    6690 distinct states found (6604 ds/min), 1405 states left on queue.
...
Progress(152) at 2014-02-09 16:24:24: 445222577 states generated (130626 s/min),
  61648071 distinct states found (8268 ds/min), 84 states left on queue.
Model checking completed. No error has been found.
  Estimates of the probability that TLC did not check all reachable states
  because two distinct states had the same fingerprint:
  calculated (optimistic):  val = .0013
  based on the actual fingerprints:  val = 3.3E-4
445223287 states generated, 61648075 distinct states found, 0 states left on queue.
The depth of the complete state graph search is 152.
Finished. (2014-02-09 16:24:52)
--------------------------------
Parsing time: 1484 ms
Translation time: 246 ms
Model checking time: 362123 sec
States analysed: 61648075
Transitions fired: 445223287
Result: NoError
```

> 100 hours

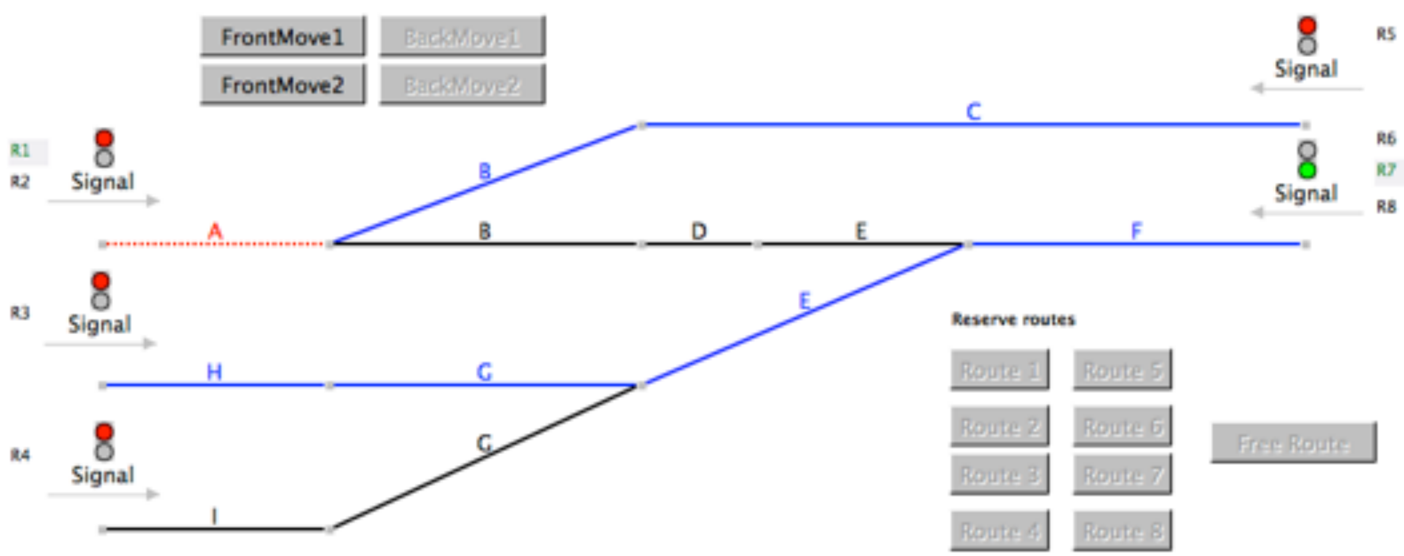
Model Checking

- The simple topology from the book certainly cannot pose any problem, can it ?

- First successful model check took **4 days**, generating **61 million** states and **445 million** transitions (we used multi core version of TLC; cf ABZ'14)



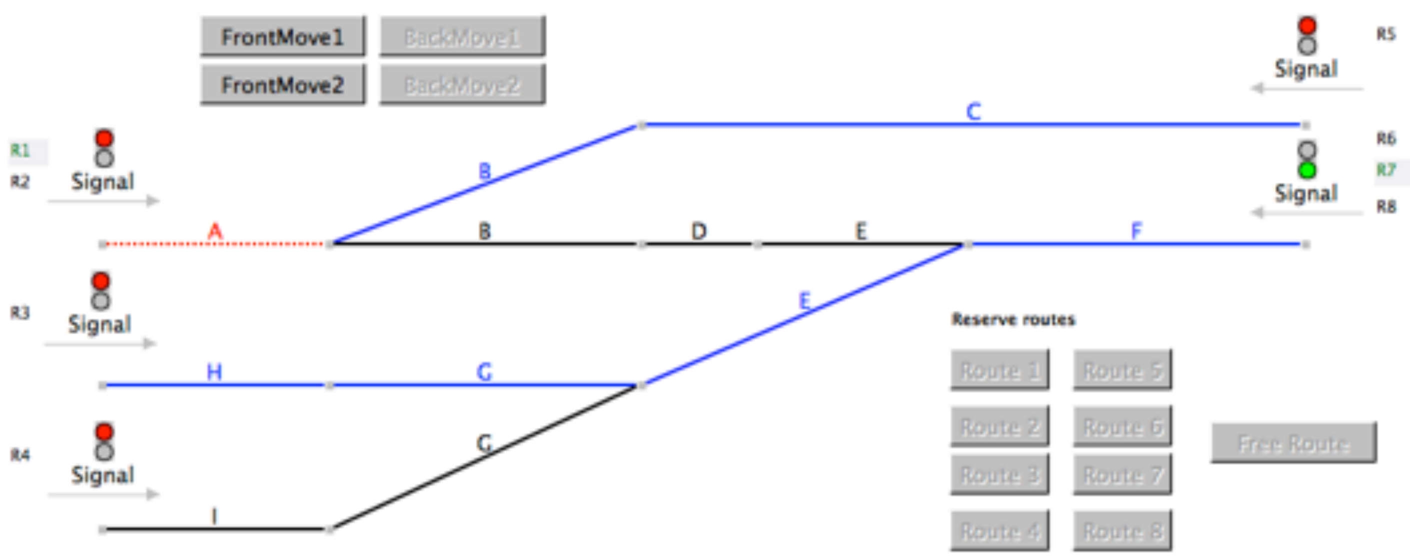
Why this blowup ??



Simpler topology: still
627,777 distinct states
(9 blocks, 5 signals, 3 points)
$2^{9+5+3} = 131{,}072$

Variables:
LBT: Last Block of Train
OCC: occupied blocks
TRK: physical layout
frm: formed routes
resbl: reserved blocks
resrt: reserved routes
resrtbl: reserved routes for reserved blocks

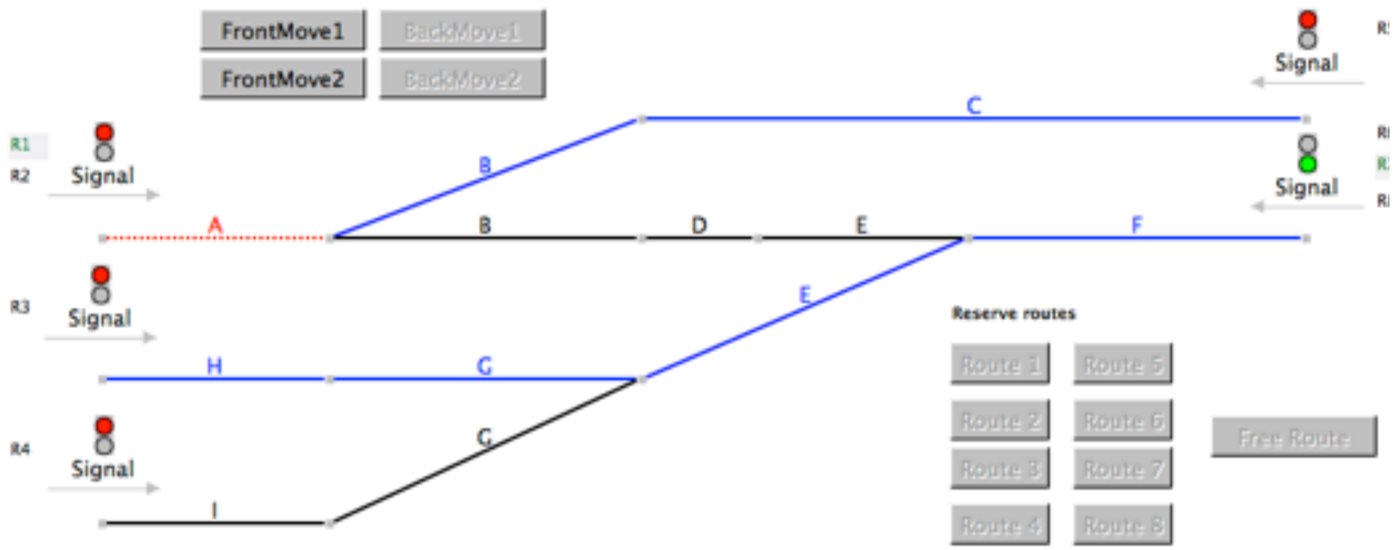
Why this blowup ??



Simpler topology: still
627,777 distinct states
(9 blocks, 5 signals, 3 points)
2^{9+5+3} = 131,072

Variables: (**Values**)
LBT: Last Block of Train **152**
OCC: occupied blocks
TRK: physical layout
frm: formed routes **256**
resbl: reserved blocks
resrt: reserved routes
resrtbl: reserved routes for reserved blocks

Why this blowup ??



Variables: (**Values**)

LBT: Last Block of Train **152**

OCC: occupied blocks

TRK: physical layout

frm: formed routes **256**

resbl: reserved blocks

resrt: reserved routes

resrtbl: reserved routes for reserved b

Coverage Table

257 Covered Values for frm (Max.Card=256)

Nr	frm	Occurences	Witness ID
232	{R1,R2,R4,R5,R7,R8}	5492	57722
233	{R1,R2,R3,R4,R5,R6,R7}	3924	199159
234	{R2,R3,R4,R7,R8}	2992	65313
235	{R1,R3,R4,R6,R7,R8}	4324	58993
236	{R2,R3,R4,R5,R6,R7}	2591	105211
237	{R7,R8}	2042	2398
238	{R1,R3,R4,R5,R7,R8}	3320	31984
239	{R1,R2,R5}	2018	11518
240	{R1,R2,R3,R4,R7,R8}	4612	72483
241	{R2,R5,R6}	1710	24472
242	{R1,R2,R6}	2200	21326
243	{R2,R3,R4,R6,R7,R8}	3688	105201
244	{R2,R3,R6,R7,R8}	3586	81134
245	{R1,R3,R5,R6,R7,R8}	4628	44416
246	{R2,R3,R4,R5,R7,R8}	3920	105202
247	{R1,R2,R3,R4,R5,R6,R8}	3924	199164
248	{R2,R3,R4,R5,R6,R8}	2591	105203
249	{R1,R2,R4,R5,R6,R7,R8}	6848	127350
250	{R2,R5}	1436	1707
251	{R1,R2,R4,R6,R7,R8}	5909	117813
252	{R1,R2,R3,R4,R5,R7,R8}	5540	155698
253	{R1,R2,R3,R4,R6,R7,R8}	6064	199011
254	{R1,R3,R4,R5,R6,R7,R8}	4772	60123
255	{R1,R2,R3,R4,R5,R6,R7,R8}	6992	199156
256	{R2,R3,R4,R5,R6,R7,R8}	4616	105196
257	{R1,R2,R3,R5,R6,R7,R8}	6848	138206

Manual POR change

- Why can all routes be formed at the same time ?
  - Because routes are not freed straightaway
- Manual “partial order” reduction: force freeing routes straightaway
  - changes model, but (maybe) we can prove that this does not hide deadlocks or invariant violations

Empirical Results

- Parallelisation very useful
- Stronger partial order reduction could be very helpful (4 days → 30 minutes)
- Latest version of TLC translation much faster now* but parB scales better: parB on Amazon 32 fastest
- model improvement was found thanks to ProB's coverage features

with POR, old TLC4B translator:

Tool	Workers	Invariant Check	Model Checking Time
TLC	1	no	69 min 11 secs
ProB	1	no	29 min 20 secs
TLC	1	yes	258 min 01 secs
TLC	2	yes	135 min 50 secs
TLC	4	yes	83 min 52 secs
TLC	8	yes	66 min 33 secs
TLC	19	yes	68 min 20 secs
ProB	1	yes	76 min 54 secs
ProB	2	yes	67 min 15 secs
ProB	4	yes	34 min 42 secs
ProB	8	yes	22 min 24 secs
ProB	19	yes	18 min 18 secs

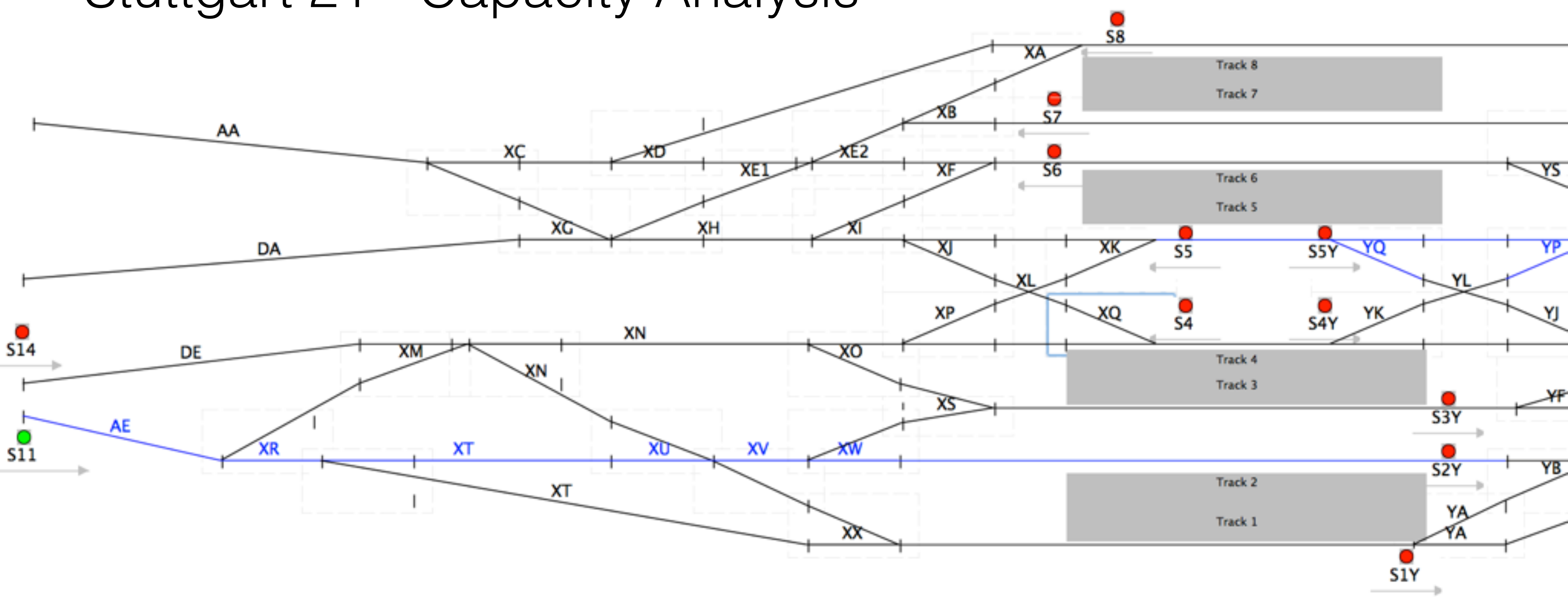
Worker	Inv	POR	TLC	States
6	no	yes	40 sec	672,173
6	yes	yes	204 sec	672,173
6	no	no	52.8 min	61,648,075
6	yes	no	305.9 min	61,648,075

* trick to avoid re-evaluation

all tests run on 6-core MacPro

Graphical Visualization For Larger Examples

Stuttgart 21 - Capacity Analysis



A more complex example

Animation with ProB, replaying Alstom test logs

cf. previous talk

Related Work

- Prover Technology (RATP): http://www.prover.com/company/casestudies/ratp/ ; iLock http://www.railway-technology.com/contractors/signal/prover-technology/
- Kirsten Winter: ISoLA 2012, Symbolic Model Checking
- Kirsten Winter et al. CSP/FDR ACSC'2003, SCS'05
- Ferrari et al. FORMS/FORMAT 2010
- …

Conclusions

- Surprising state explosion of interlocking model
- Coverage analysis tools useful to diagnose state space explosion
  - reached values for single variable, for all variables (all values or just minimum and maximum)
- We are still far away from scaling exhaustive model checking to a realistic interlocking; stronger partial order reduction could help (papers serves as reference point for other approaches)
- Proof or combination of proof, modelling and model checking
- Animation/constraint solving was useful in finding “unexpected” behaviour in fully proven model