University of Leicester

Modular Stochastic Graph Transformation

Reiko Heckel, University of Leicester

(joint work with G. Laijos and S. Menge, Dortmund)

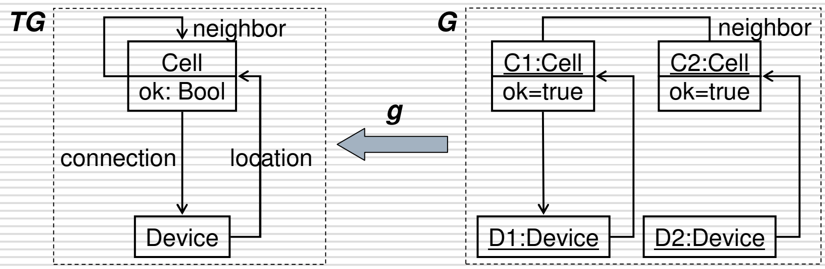
IFIP WG 1.3 Meeting
Bangalore, January 2005

Motivation: Modelling and Analyzing Mobile Systems

- mobility is naturally modelled in graph transformation systems, see e.g.
  - semantics of pi and ambient calculus
  - applications to model individual systems (or styles)
- non-functional quality aspects (availability, MTBF, response time, ...)
  - require analysis methods with probabilistic time
  - are most critical in mobile systems (or generally, for systems with dynamic network topology)

University of Leicester

Typed Graph Transformation System: Nomadic Wireless Network



- skip edge labels where convenient
- show pairs of opposite edges as undirected

Graph Transformation System

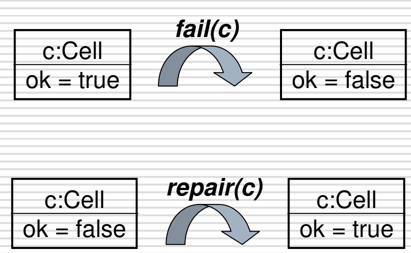
GTS **G** = ⟨ **TG**, **P**, **R**, $\mathbf{G_0}$ ⟩

- type graph **TG**
- rule names $\mathbf{P = (P_w)}_{w \in |TG|}$
- rules **R** of the form

  $\mathbf{p(x_1, \ldots, x_n): L \rightarrow R}$

  with formal params $\mathbf{x_i \in L}$ **or R** typed over **TG**
- start graph $\mathbf{G_0}$ typed over **TG**

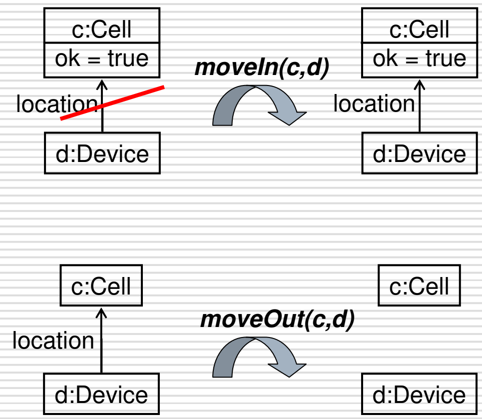
Transformation G $\rightarrow_{p(o)}$ H
acc. to DPO approach.



- Time between failures unknown, but much longer on average than time to repair.

Moving In and Out of Range

- moving is change of location
- moving in and out should be infrequent

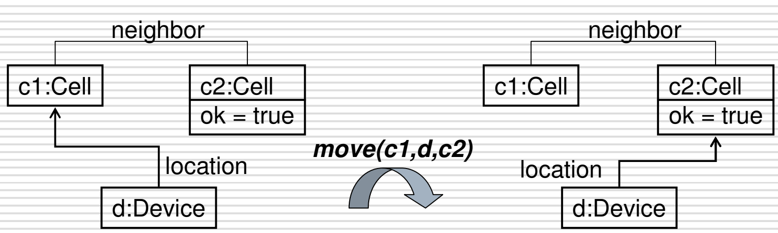


University of Leicester

More Moving

- if coverage is good, we should move between neighbouring stations without leaving the network
- effect is the same as of

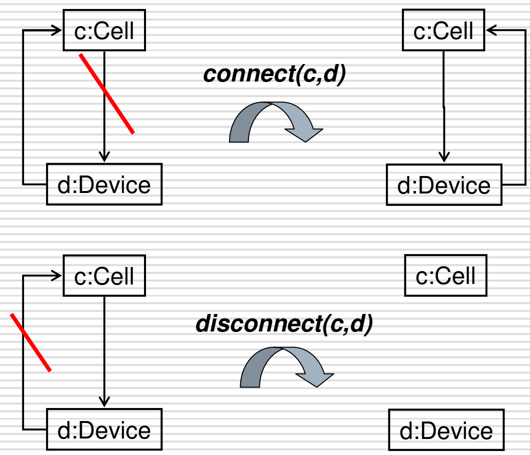
  moveOut(c1, d); moveIn(c2, d)
- (hopefully) more frequent



University of Leicester

Connect and Disconnect

- if device is located in a cell and has no connection, one is established
- if device is no longer in the cell it is connected to, the connection is lost
- should both be faster (when enabled) than moving or repairing



Approach: Following the Crowd (e.g., Stochastic Petri Nets)

- use state-based formalism to generate labelled transition system

  $$LTS = \langle S, L, \rightarrow \rangle$$

- associate labels with rates of exponential probability distribution

  $$\rho: L \rightarrow R^+$$

- derive cont.-time Markov chain (CTMC), an $S \times S$ matrix $Q=Q(s,s')_{s,s' \in S}$, by

  $$Q(s,s') = \Sigma_{(l,\, s') \in s \rightarrow}\ \rho(l)$$

  where $s\rightarrow \; = \{ (l, s') \mid s \rightarrow_l s' \}$

Graph Transition System generated by GTS

LTS(G) = ⟨ S, L, → ⟩ where

- S is set of *standard representations* of graphs reachable from G_0
- L is set of all rule names with (actual) parameters
- → ⊆ S × L × S is transfomation relation, i.e. $G \rightarrow_{p(a1, ..., an)} H$ *iff* exists trafo via $p(x_1, ... x_n)$ compatible with assignment $x_i \rightarrow a_i$

University of Leicester

Stochastic Graph Transformation System

A stochastic GTS SG = ⟨ TG, P, R, G_0 , ρ ⟩ consists of

- a GTS ⟨ TG, P, R , G_0 ⟩
- a mapping ρ: P → R^+ associating with each p its rate ρ(p)

rule name p	rate ρ(p)	rule name p	rate ρ(p)
repair	500	fail	1
moveIn	1	moveOut	1
move	100		
connect	10000	disconnect	10000

read: 100 × per day
(= 4 min avg. delay)

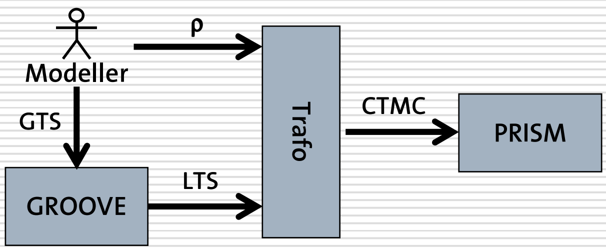
University of Leicester

Experimental Tool Chain

To answer questions like

What is, in the long run, the probability for a given device to have a connection?

- generate LTS from GTS and supply rates in separate file
- generate CTMC + evaluation of atomic propositions for stochastic model checker PRISM



Problem: generation of transition systems

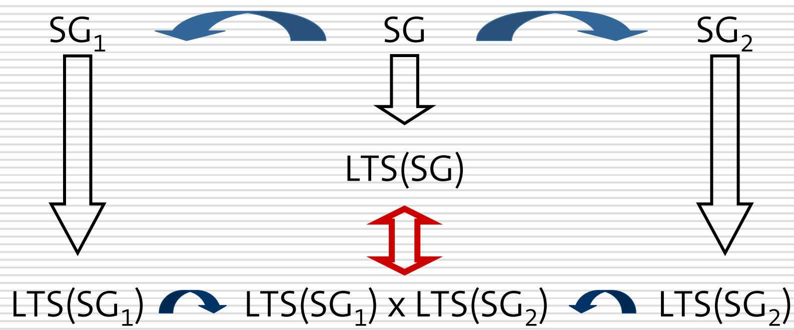
University of Leicester

Outline

- Stochastic Graph Transformation
  - ✓ modelling
  - ✓ analysis
  - modularisation

University of Leicester

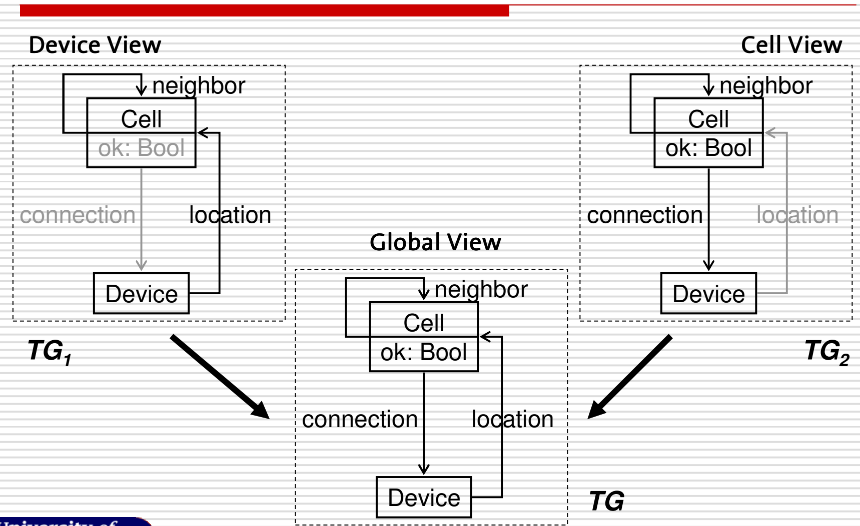
Modular Generation



- decomposition: projection of graphs, rules, and GTS
- composition
- equivalence ?

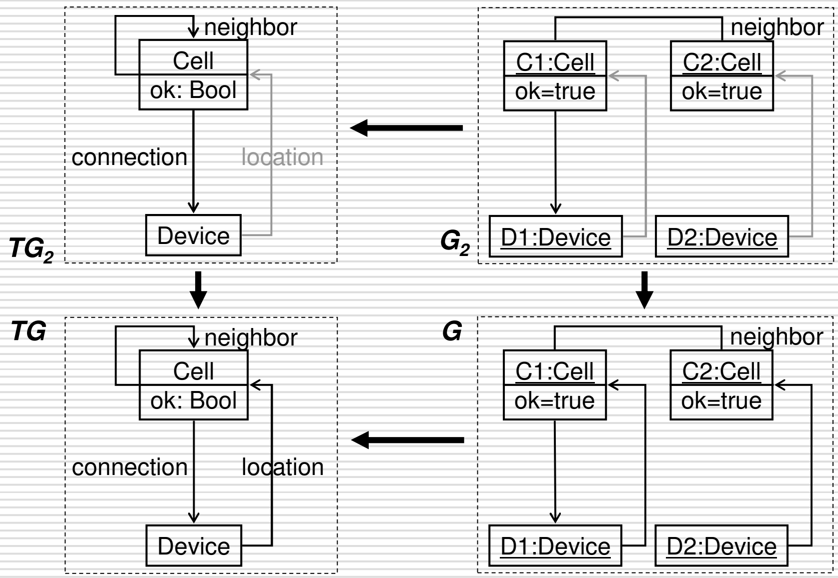
University *of* Leicester

Decomposition of Type Graphs ...

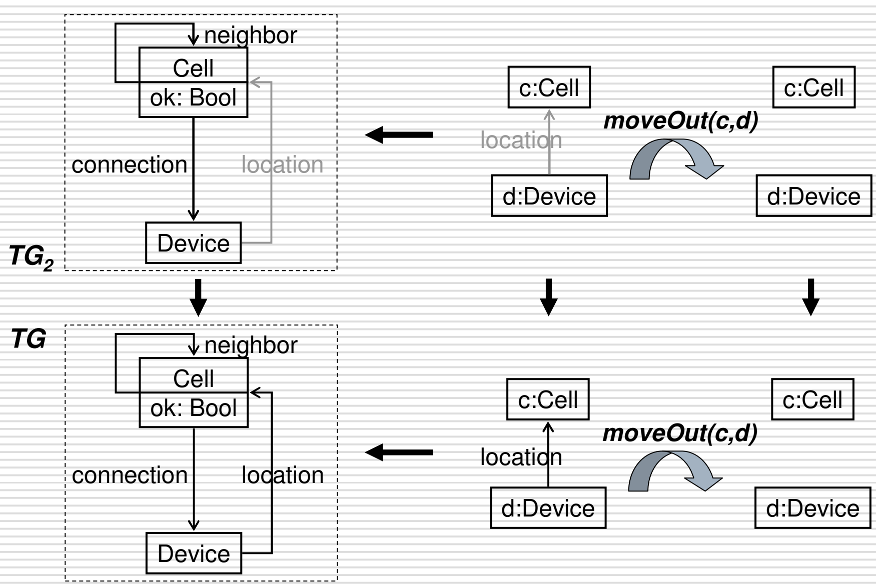


University *of* Leicester

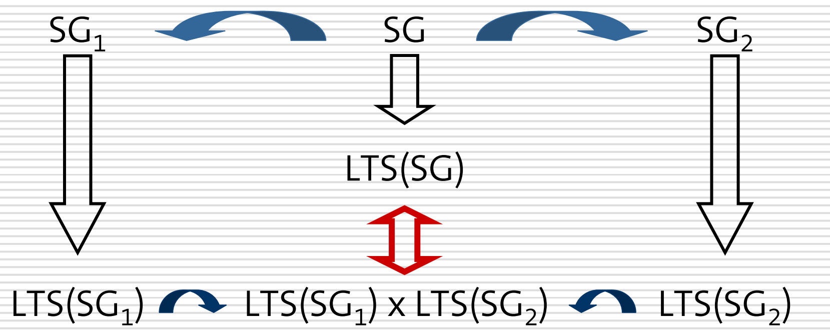
… yields Projection of Instance Graphs,



… Rules, and Transformation Systems



Modular Generation



- decomposition: projection of graphs, rules, and GTS
- composition: product with synchronization on common labels
- equivalence: make sure only *compatible* transitions are synchronized

University *of* Leicester

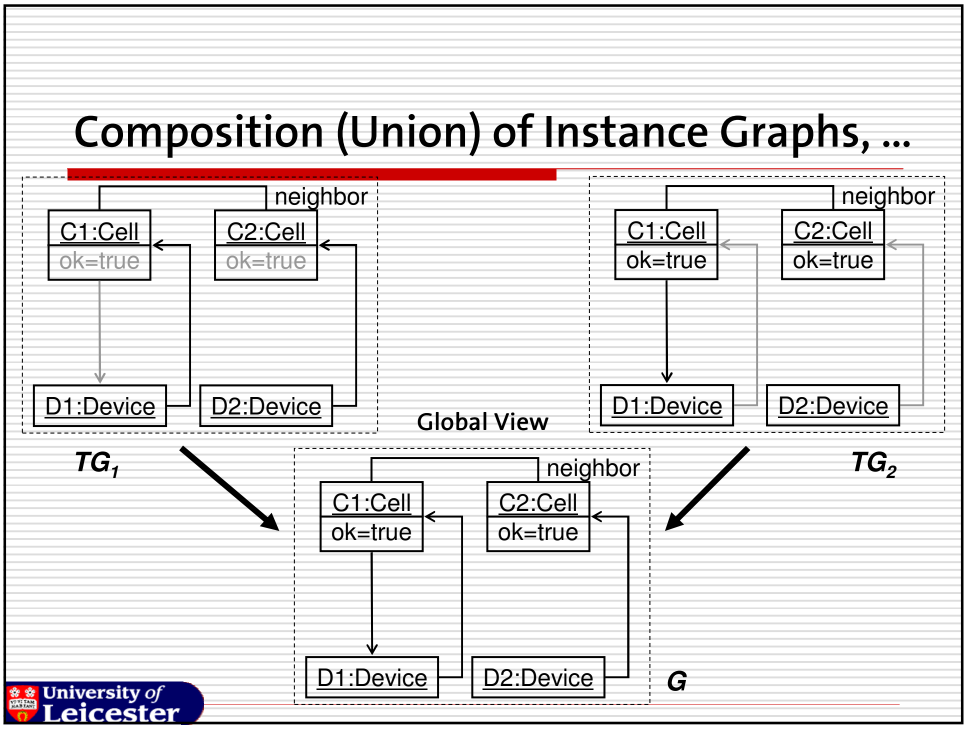
Compatibility

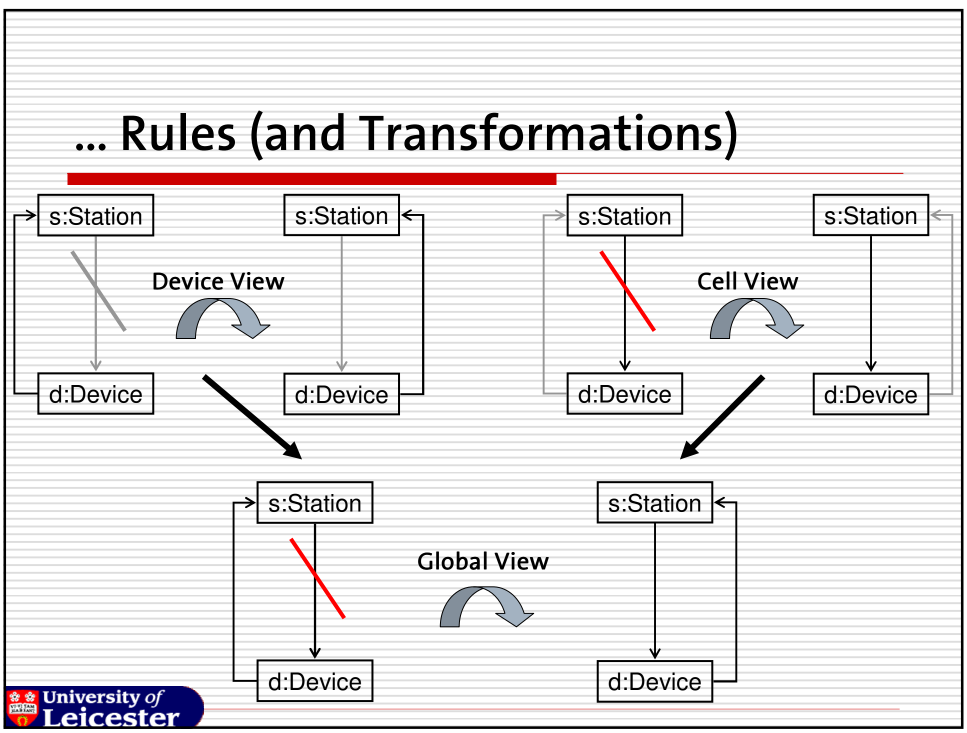
- graphs are compatible if their union is well-defined: $G = G_1 \cup G_2$
- two steps are compatible if
  - G_1 and G_2 are compatible
  - r_1 and r_2 are compatible: $r = r_1 \cup r_2$
  - occurrences are compatible (agree on $L_1 \cap L_2$): $o = o_1 \cap o_2$

IFF ex. global step $G \rightarrow_{r(o)} H = H_1 \cup H_2$

[Distribution Thm; Ehrig, Taentzer]

University *of* Leicester


Composition (Union) of Instance Graphs, ...
neighbor
C1:Cell
ok=true
C2:Cell
ok=true
D1:Device
D2:Device
TG1
Global View
neighbor
C1:Cell
ok=true
C2:Cell
ok=true
D1:Device
D2:Device
TG2
neighbor
C1:Cell
ok=true
C2:Cell
ok=true
D1:Device
D2:Device
G
University of Leicester


... Rules (and Transformations)
s:Station
s:Station
Device View
d:Device
d:Device
s:Station
s:Station
Cell View
d:Device
d:Device
s:Station
Global View
s:Station
d:Device
d:Device
University of Leicester

Compositionality

Behaviour of projected GTS, subsequently composed, is bisimilar to the original one.

$$LTS(G) \sim LTS(G|_{TG1}) \times LTS(G|_{TG2})$$

... provided that *equality on labels* coincides with *compatibility of steps:*

- product of LTS controlled by labels: set up labels such that

  $G_1 \Rightarrow_{p1(a1)} H_1$ and $G_2 \Rightarrow_{p2(a2)} H_2$

  are compatible iff $p_1(o_1) = p_2(o_2)$, i.e.,
  - corresponding rules in different views have same name and parameters
  - parameters completely determine occurrence on $L_1 \cap L_2$ so that steps with the same label agree on changes on common parts
- labels reflect the interaction of views

Reduction

- (Somewhat) larger systems can be analysed

Number of cells/devices	No of States Device View	No of States Cell view	No of States Global View
2/2	51	76	636
3/2	219	430	10090
3/3	11687	8240	out of memory

Summary and Open Questions

- intuitive visual modelling + standard stoch. analysis
  - another convenient way to generate complex LTS
- experimental tool support
- compositional generation
  - somewhat more difficult since GT are not compositional *per se*
- more general models
  - instantaneous and external steps
  - true concurrency
- co-algebraic presentation
  - link between notions of *system* and *logic*
  - internal co-algebras capturing compositional behaviour

University *of* Leicester