Introducing the CoSta Project: Contractual Statecharts

Gerald Lüttgen

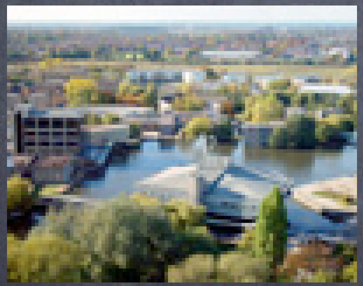
High Integrity Systems Engineering Research Group &
Programming Languages and Systems Research Group

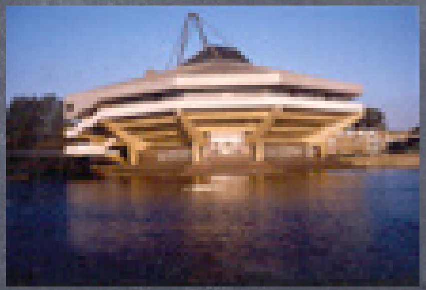
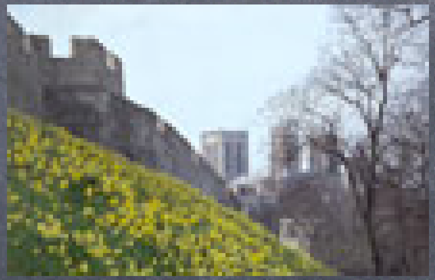
Department of Computer Science
University of York, U.K.

Talk at SYNCHRON, Bamberg, Germany, November 2007.

ETAPS 2009 in York: 22^{nd}-29^{th} March

- 12th European Joint Conferences on Theory and Practice of Software
  - Primary forum for academic and industrial researchers working on topics relating to Software Science
  - Confederation of five main annual conferences, accompanied by satellite workshops and other events




- Run a satellite event - a workshop or a tutorial!
  - See www.cs.york.ac.uk/etaps09/ for details on how to propose satellites (deadline: 14^{th} January 2008)

The CoSta Research Project

- Three-year EPSRC-funded research project at York:
  - BAE Systems as industrial collaborator
  - Research team consists of GL (PI), Dr Richard Paige (Co-PI), Dr Andy Galloway (RA) and Ms Lishan Harbird (RS)
- Strategic goal: Improve the foundations and tool support for designing and building avionics software
- Observation: Statechart languages are not fully adequate for early design stages and refinement-based design
  - No support for declarative styles of specification
  - No adequate facilities for component-based design

Application Domain: Avionics Software

- Project idea is the result of stimulating discussions at NASA LaRC on designs of future flight control systems
- Software engineers in avionics routinely rely on decades of experience with avionics software and architectures
  - Design starts from existing architecture, plus requirements
  - Design finishes with executable model (validated via simulation)
- Different specification styles are appropriate at different design stages
  - Requirements predominently have a declarative character
  - Concrete designs are typically operational

The Bigger Picture

- In practice, software engineers typically mix different styles of specification:
  - Design languages, such as the UML, combine state machines with the declarative Object Contraint Language (OCL)
  - Programming languages, such as Eiffel, combine imperative language constructs with declarative contracts (assume-guarantee-reasoning)
- In formal methods, the focus is on pure theories
  - Operational theories - automata theory, process algebras, ...
  - Declarative theories - set-based notations, temporal logics, ...

thus often ignoring the "heterogeneous" reality!

The CoSta Project: Objectives

- Extending Statecharts (Stateflow) by contracts (temporal safety properties) so as to support mixed operational and declarative specification styles

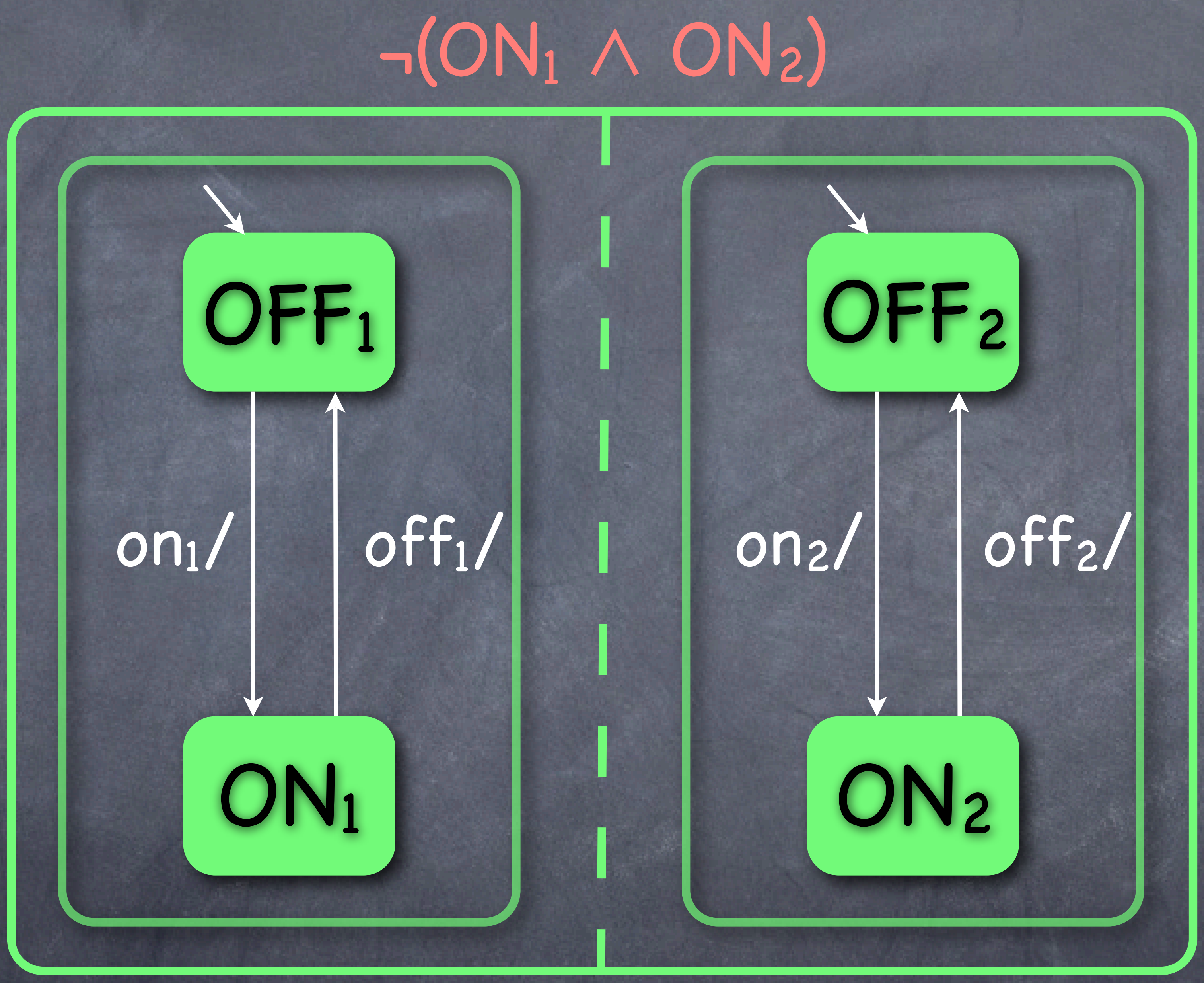
- Developing a refinement relation for component-based stepwise design that permits one to trade off declarative content for operational content

- Driven by industrial case studies, provided by our project partner BAE Systems

- Tool support in form of a simulator and a model checker

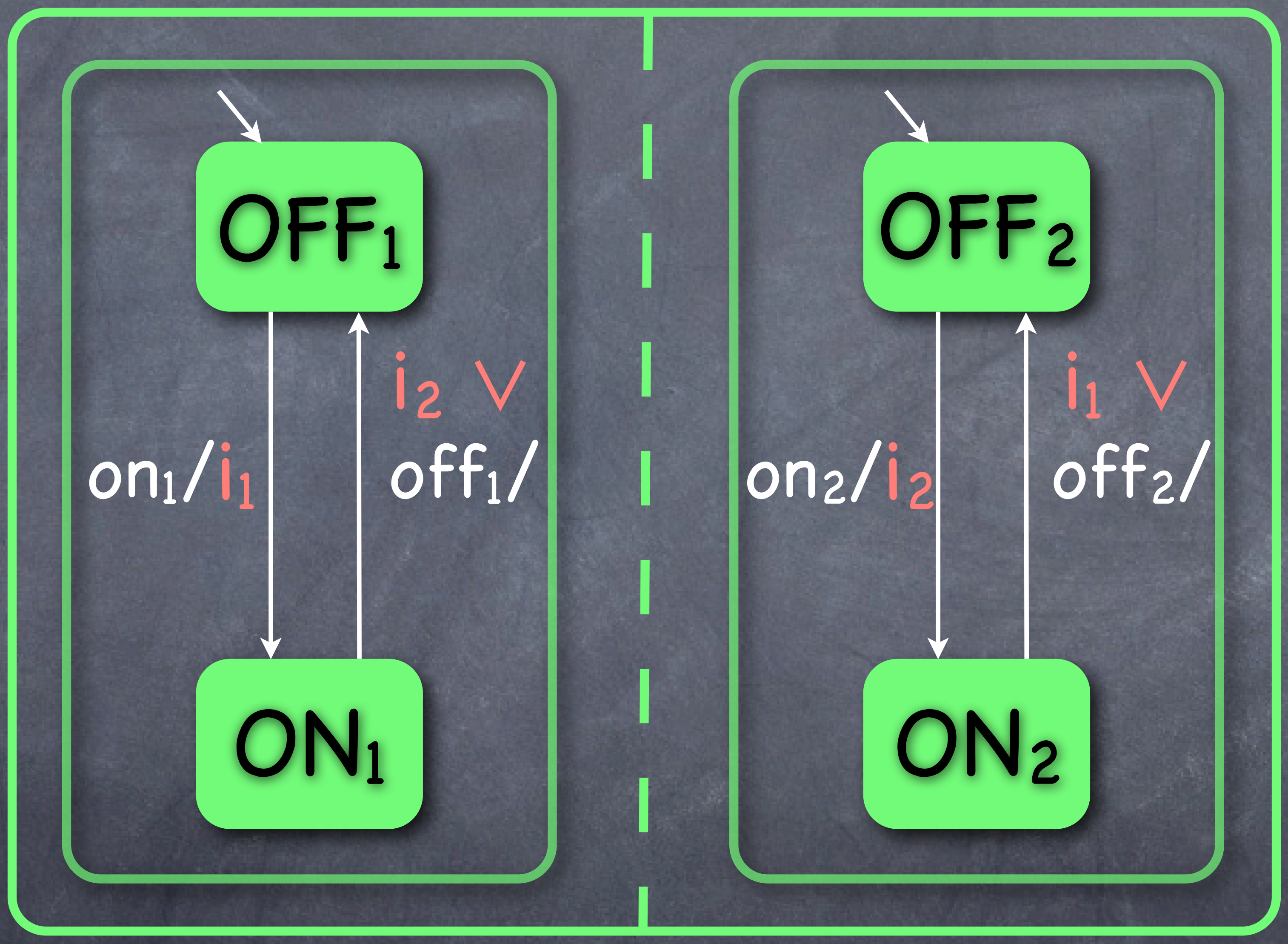
- Refinement patterns that capture standard rules for translating between operational and declarative content

Refinement Patterns: A Mode-Logic Example



Mutually-exclusive states via contract

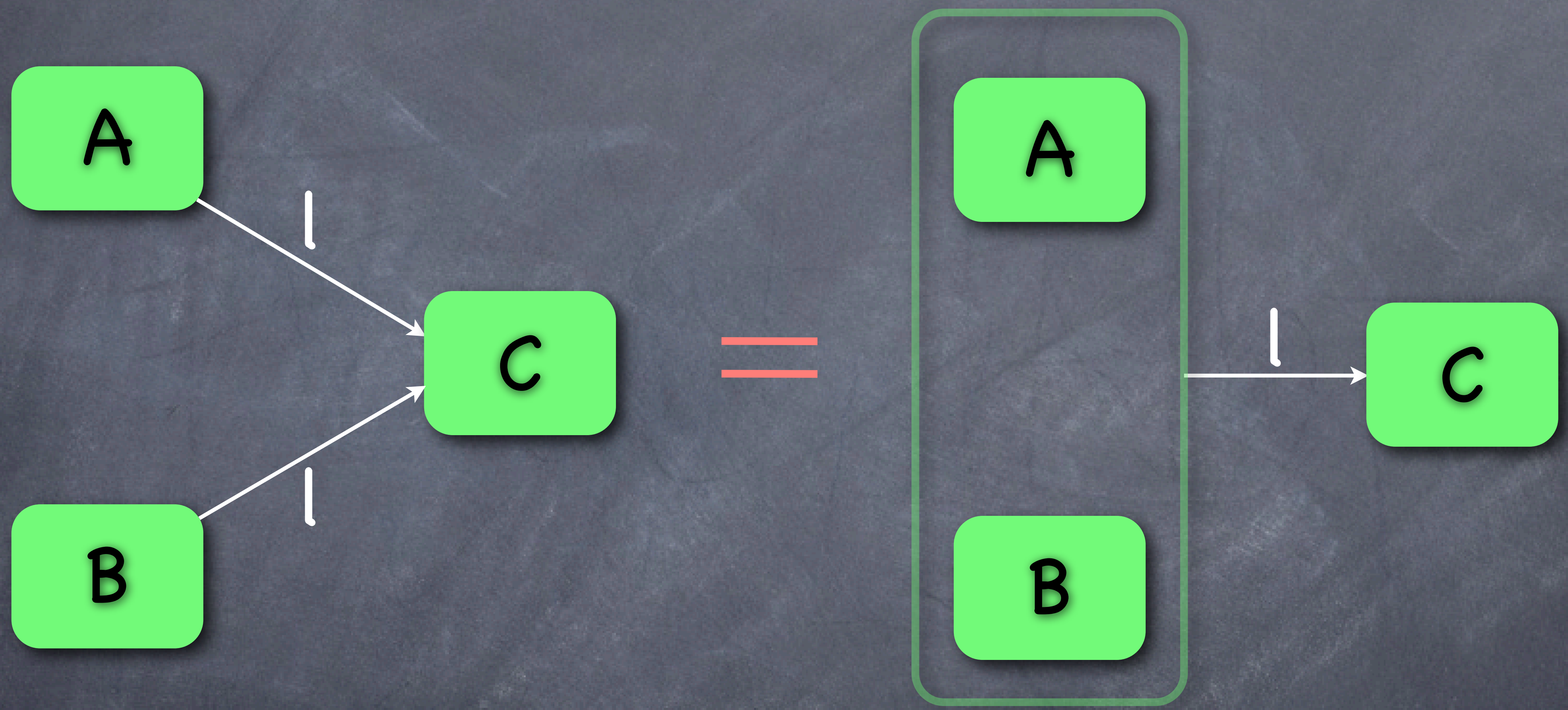
Refinement Patterns: A Mode-Logic Example



Mutually-exclusive states via event broadcasting

Other Patterns: Equivalence Patterns

(cf. KIEL's Layouter [Reinhard von Hanxleden et al])



(Assumes semantics of or-states without implicit priority)

Concrete Questions to be Investigated

- How exactly to enrich Statecharts with contracts?
- What should the contract language be?
  - Pre-/post-conditions and invariants on states and transitions, temporal safety properties, ...
- Which semantics and behavioural preorders are suitable?
  - Compositionality is mandatory since refinement patterns demand an open-systems view
  - Refinement should permit the resolution of disjunctive choices
- Which refinement patterns are applied in avionics?
  - How to formalise refinement patterns?

Some Related Work

- Logic-time contracts for reactive embedded components
  - The CoSta contract language shall be a first-class citizen within the mixed design notation
- Extending OCL with temporal logics inside UML
  - Specifying global and local invariants between objects, and pre-/post-conditions of methods
  - Designing avionics software with the UML?
- Design patterns for programming languages
  - Focusing on transforming designs to implementations, rather than on refining high-level declarative designs to low-level operational designs

Foundations of CoSta's Semantic Backbone

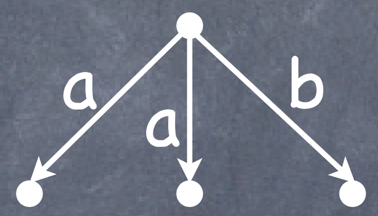
[Joint work with Walter Vogler at FOSSACS'06 & ICALP'07]

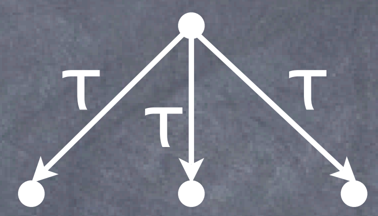
- Logic Labelled Transition Systems
  - Inconsistency as an observable entity
- Composition operators on Logic LTS
  - Parallel operator, conjunction, disjunction, temporal operators
- Two fully-abstact refinement preorders
  - "Synchronous setting": Fully-synchronous parallel composition
  - "Asynchronous setting": CSP-style parallel composition
- Logic properties of these behavioural preorders
  - $\wedge$ is conjunction, distributivity laws, ...

The Setting of Logic LTS

LTS over alphabet that includes the silent event τ, plus:

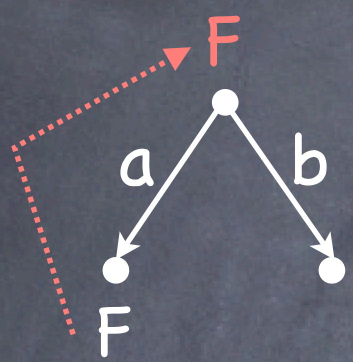
- τ-purity, i.e., each state encodes either external choice or internal choice



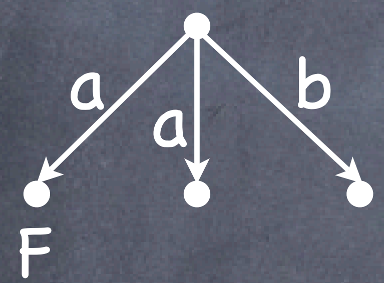


- Inconsistency predicate F on states
  - Inconsistencies can arise by conjunctive composition
  - Runs through inconsistent states are semantically filtered out
  - Inconsistencies can propagate backwards along transitions ...

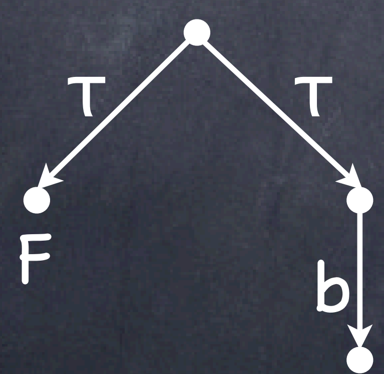
Backward Propagation of Inconsistencies



Propagation - If the environment insists on performing a, the process is forced to enter the inconsistent state



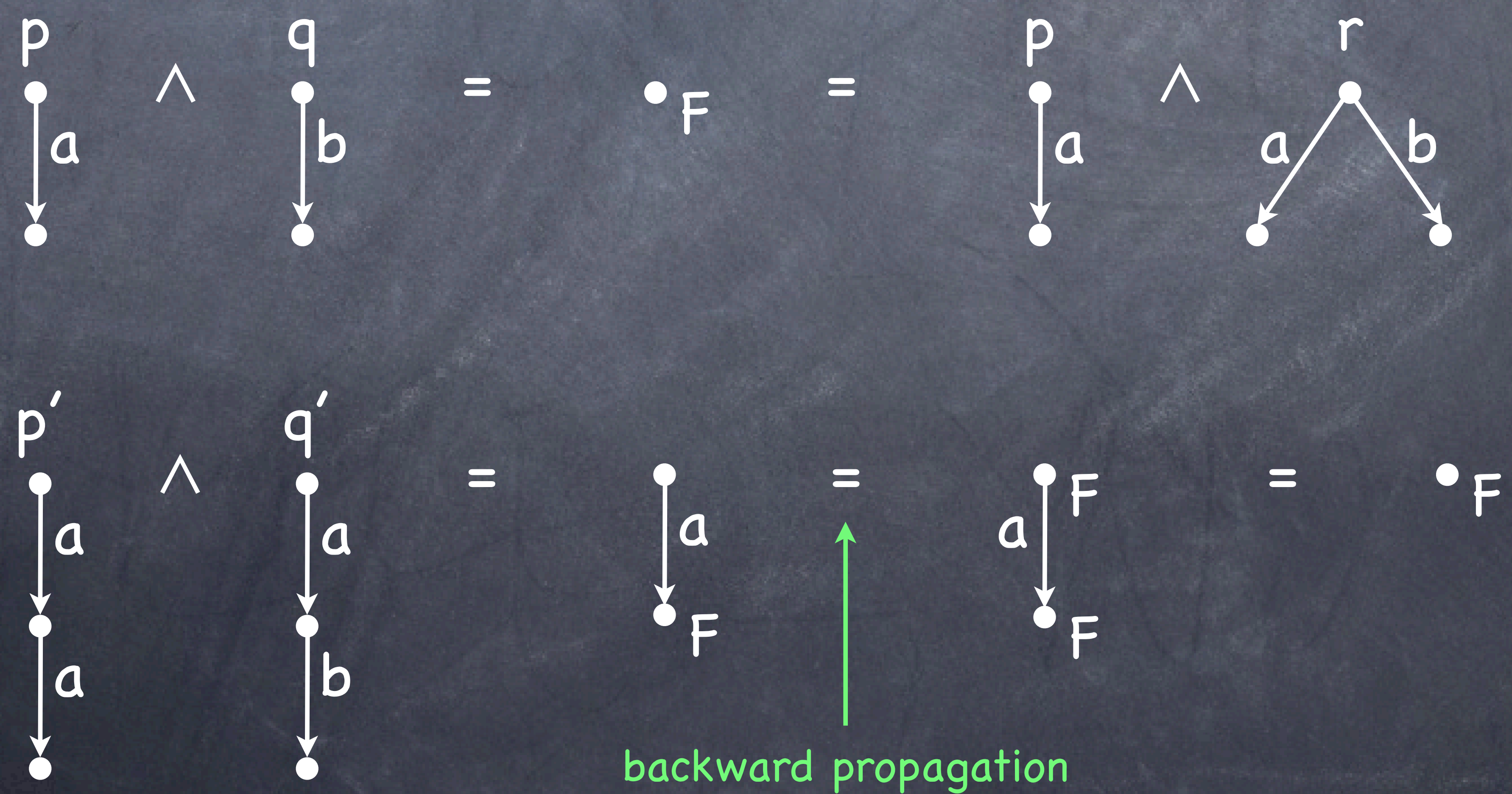
No propagation - While the environment can insist on a, the process can decide to perform the "good a"



No propagation - The process decides on its own which τ-branch to follow ("disjunction")

Conjunction on Logic LTS

- Synchronous composition, but considering inconsistencies
- Inconsistency ⇔ different ready sets, i.e., if one process offers an event that the other cannot perform
- Examples:



Synchronous Product and Conjunction

- Why not simply define conjunction as the ordinary synchronous product on standard LTS?
- Given a refinement preorder $\leq$, a conjunction operator $\wedge$ should satisfy:

  $r \leq p \wedge q$ if and only if $r \leq p$ and $r \leq q$

- When taking $\wedge$ to be the synchronous product:

  $0 \leq a \wedge b$ but neither $0 \leq a$ nor $0 \leq b$

  for any reasonable $\leq$, where 0 stands for deadlock

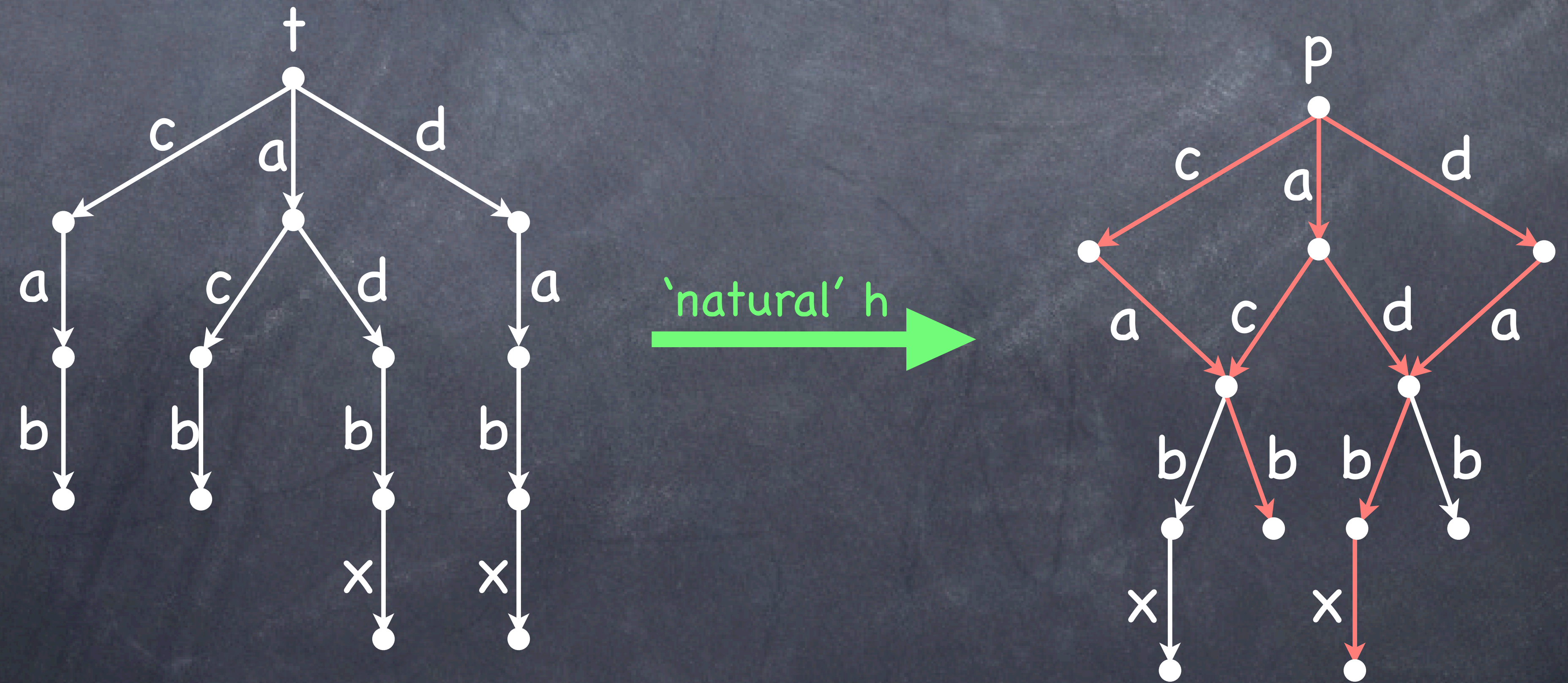
- Hence: differentiate between deadlock and inconsistency!

[Divergence]

Ready-Tree Semantics

(cf. Possible-Worlds Semantics of [Veglioni/De Nicola, van Glabbeek, 1998])

- Ready tree t of LTS p
  - Deterministic, tree-shaped LTS without τ's (stable states only)
  - Mapping h from states of t to stable states of p, which must preserve ready sets
- Example:



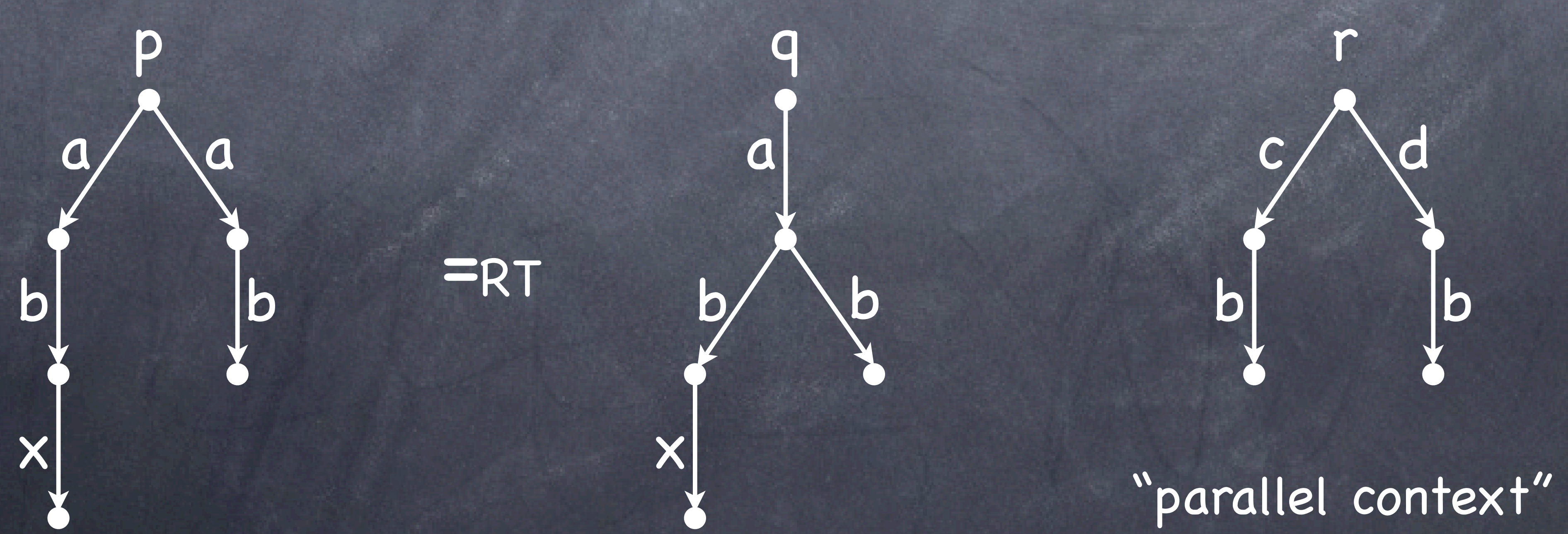
Full Abstraction wrt. Conjunction

- Ready-tree preorder:
  - $p \leq_{RT} q$ if $\forall t.$ t is ready tree of $p \Rightarrow t$ is ready tree of q
  - Lies between failures inclusion and ready simulation
- Inconsistency preorder (as reference point):
  - $p \leq_F q$ if p consistent $\Rightarrow$ q consistent
    - A consistent implementation p does never refine an inconsistent specification q ("inconsistent requirements can never be satisfied")
- Full-abstraction result:
  - $\leq_{RT}$ is the largest precongruence wrt. $\wedge$ in $\leq_F$, i.e.,

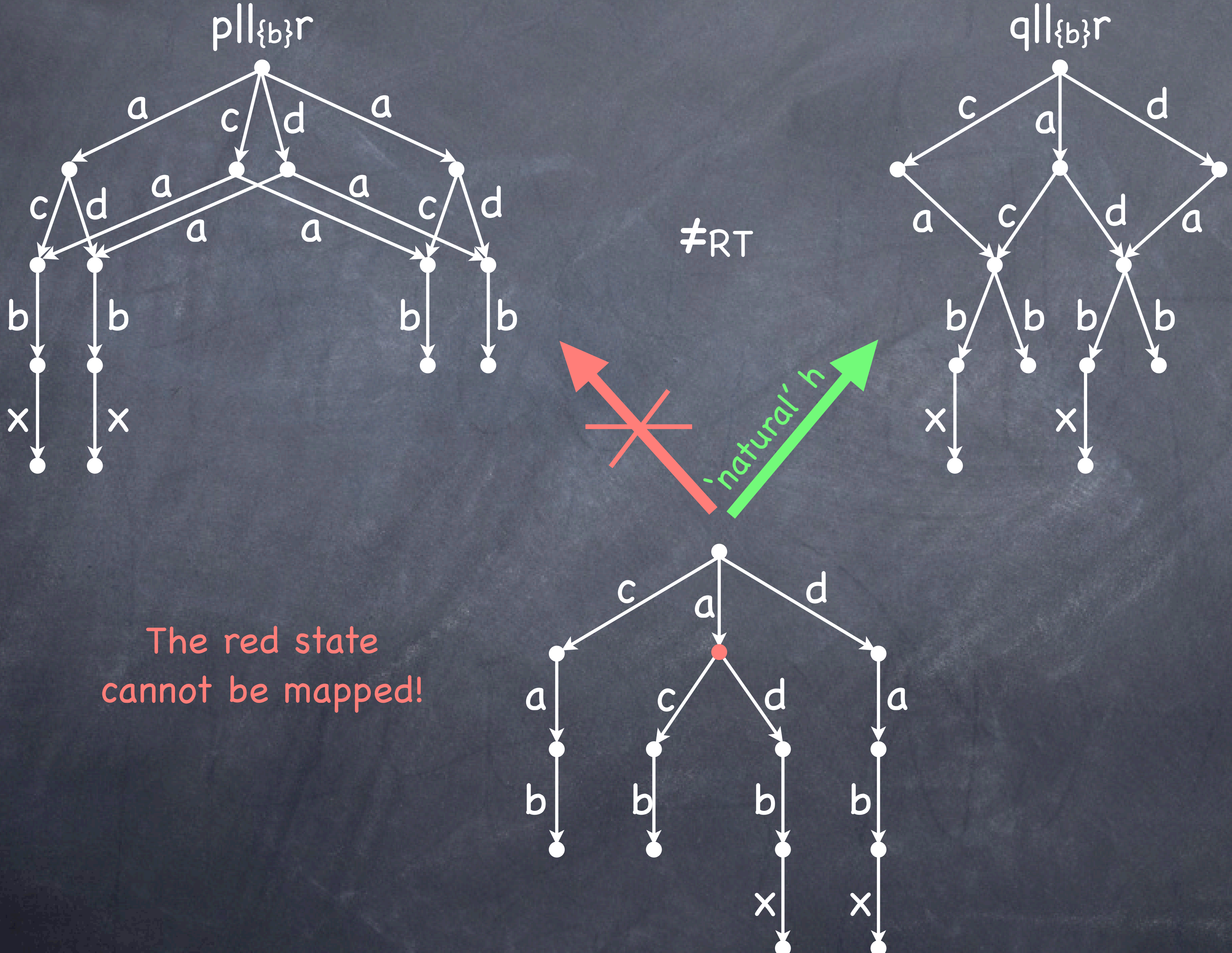
    $p \leq_{RT} q$ if and only if $\forall r.\ p \wedge r \leq_F q \wedge r$

What about Parallel Composition on Logic LTS?

- Good news: Ready-tree preorder is also compositional for the fully synchronous product ||
- Bad news: Ready-tree preorder is NOT compositional for CSP-style parallel composition $||_A$
- Compositionality defect illustrated:



Compositionality Defect Illustrated



Ready Simulation & Full Abstraction

- Adaptation of ready simulation [Bloom/Istrail/Meyer, 1988] to Logic LTS, i.e., $p \leq_{RS} q$ if
  - Consistent steps of p can be matched by consistent steps of q
  - Stable states of p are matched by stable states of q that offer the same ready set
- Full-abstraction result:
  - $\leq_{RS}$ is the largest precongruence wrt. $\wedge$ and $\|_A$ in $\leq_F$
  - It suffices in the proof to relate $\leq_{RS}$ to $\leq_{RT}$, given the previous full-abstraction result [details in ICALP'07 paper]

[Details]

Logic Properties of Both Preorders

- $\wedge$ is 'and':
  - $r \leq p \wedge q$ if and only if $r \leq p$ and $r \leq q$
  - Note again that this does not hold if $\wedge$ is simply taken to be the synchronous product
- Further properties:
  - $p \wedge q = p$ if and only if $p \leq q$
  - $p \wedge q \leq p$
  - $p \wedge p = p$
  - $p \wedge ff = ff$ (ff is Logic LTS with a single, inconsistent state)

Extensions to Other Desired Operators

- Standard logic operators:
  - Disjunction – $p \vee q$ "internal choice"
  - Negation on events – $\neg a$
- Temporal operators ("safety properties"):
  - Always – $\Box\, p$ "p holds in every step/state"
  - Bounded eventually – $\Diamond^{\leq k}\, p$ "p holds within k steps"
- Embedding of temporal logic formulas is conservative:
  - p sat φ if and only if $p \leq \varphi$

Next Milestone for the CoSta Project

(Progress report at SYNCHRON'08)

- Design and implementation of the envisaged "Stateflow + Contracts" language
- Adaptation of the Logic LTS framework to this language
- Required modifications to the Logic LTS framework:
  - Adapt transition labels to input/output-style labels
  - Integrate shared variables - second communication mechanism
  - Add a "true" predicate T - full underspecification
- Desired - but probably a long-term goal:
  - Extend framework to support the synchrony hypothesis

Thank You!

- Questions?
- Some selected references:

- G. Lüttgen and W. Vogler. Conjunction on processes: Full-abstraction via ready-tree semantics. TCS 373(1-2):19-40, 2007.
- G. Lüttgen and W. Vogler. Ready simulation for concurrency: It's logical! In ICALP, LNCS 4596:752-763, Springer, 2007.
- F. Maraninchi and L. Morel. Logic-time contracts for reactive embedded components. In EUROMICRO, pp.48-55. IEEE Press, 2004.
- E.-R. Olderog. Nets, terms and formulas. Cambridge Tracts in Theoretical Computer Science 23. Cambridge University Press, 1991.