Ready Simulation for Concurrency: It's Logical!

Gerald Luettgen
Department of Computer Science
University of York

Talk at the School of Informatics, University of Edinburgh, September 2007.

©2007 Gerald Luettgen

Menu

Appetiser

Brief overview of my current research

Main Course

Concurrency-theoretic foundations of heterogeneous design notations

Desert

Potential synergies with SICSA themes

Appetiser: Overview of My Current Research

- Concurrency theory:
  - Observation - gap between software engineering practice and mathematical foundations
    - Practice: Mix of different specification/design styles; e.g., UML combines state machines (operational style) and OCL (declarative)
    - Foundations: Investigation of pure theories, e.g., process algebras and temporal logics
  - Current EPSRC-funded research project, with BAE Systems and Rance Cleaveland (U. Maryland, USA) as collaborators
    - A calculus that combines process-algebraic and logic operators, together with a refinement-based semantic theory
    - Stateflow with "contracts" (temporal safety properties), equipped with "refinement patterns" (inequational laws)

Further Active Research Areas

- Automated verification:
  - Efficient symbolic model checking for asynchronous systems [FMSD 31(1), TACAS'07, ICATPN'07]
  - Parallelising such model checkers on multi-core PCs [CAV'07, PDMC'07]
  - EPSRC funded; collaborators Gianfranco Ciardo (UC Riverside, USA) and Radu Siminiceanu (NASA/NIA, USA)
- Synchronous languages (Statecharts, Esterel, etc.):
  - First fully abstract semantics, based on intuitionistic logic [ACM TOCL 3(1)]
  - Joint work with Michael Mendler (U. Bamberg, D)

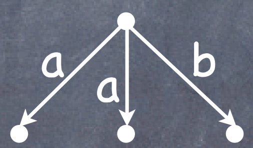
Main Course

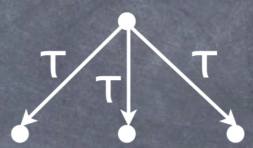
- Long-term goal:
  - Mixing process algebras and temporal logics in a single refinement-based theory
- This talk:
  - Presents the setting of Logic Labelled Transition Systems
  - Shows that ready simulation is fully abstract when considering both conjunction and parallel composition on Logic LTS
  - Investigates logic properties of ready simulation
  - Reports on joint work with Walter Vogler (U. Augsburg, D); details in [TCS 373(1-2), FOSSACS'06, ICALP'07]

Setting - Logic LTS

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

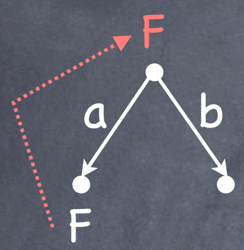
- τ-purity
  - Each state encodes either external choice or internal choice



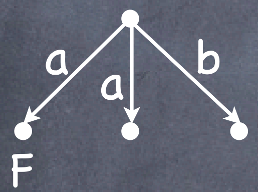


- Inconsistency predicate F on states
  - Inconsistencies can arise by conjunctive composition
  - Runs through inconsistent states are semantically filtered out
  - Inconsistencies can propagate backwards along transitions ...

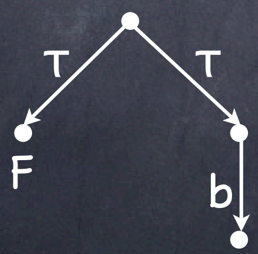
Backward Propagation of Inconsistencies



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



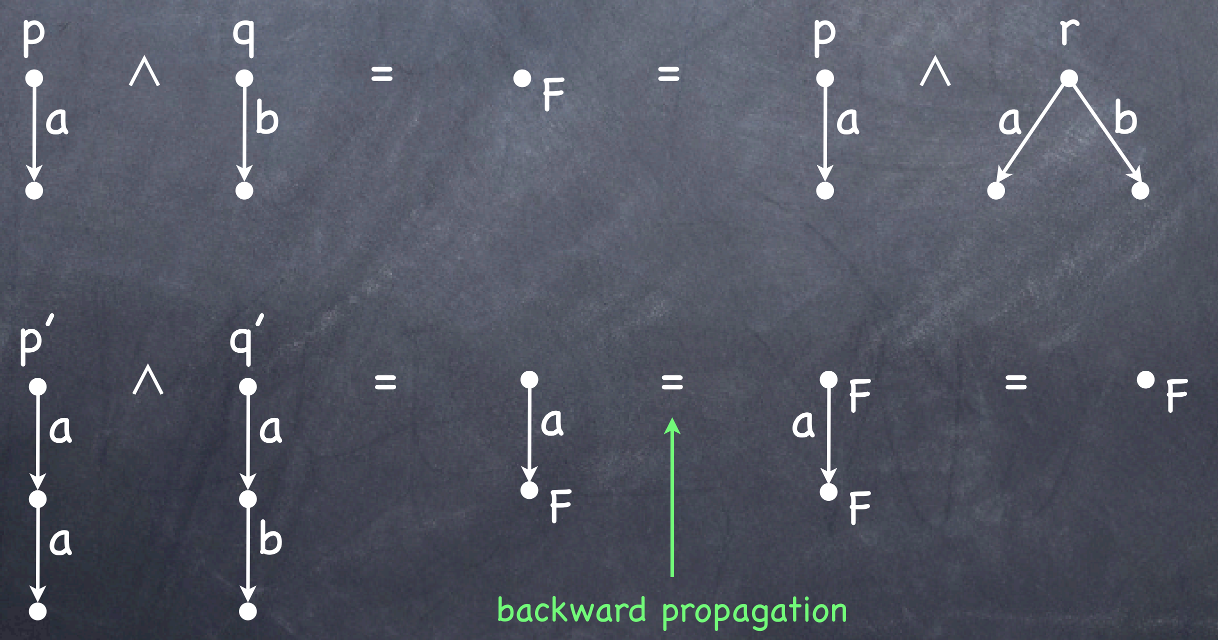
No propagation – While the environment can insist on a, the process can decide to perform the "good a"



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

Conjunction on Logic LTS

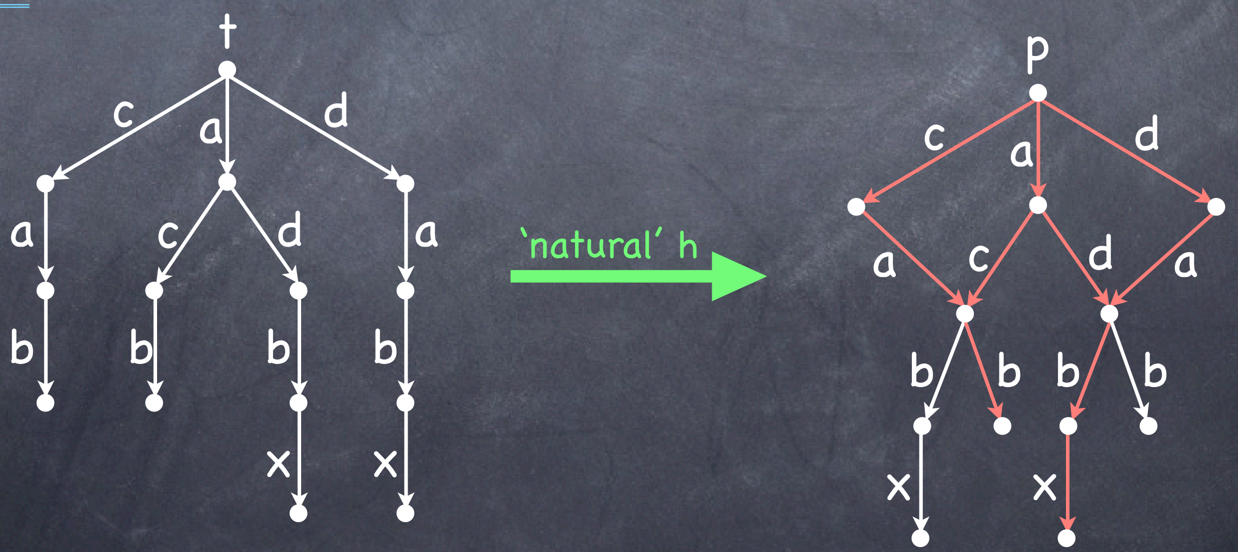
- Synchronous composition, but considering inconsistencies
- Inconsistency ⇔ different ready sets, i.e., if one process offers an action that the other cannot perform
- Examples:



Ready-Tree Semantics

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

- 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:



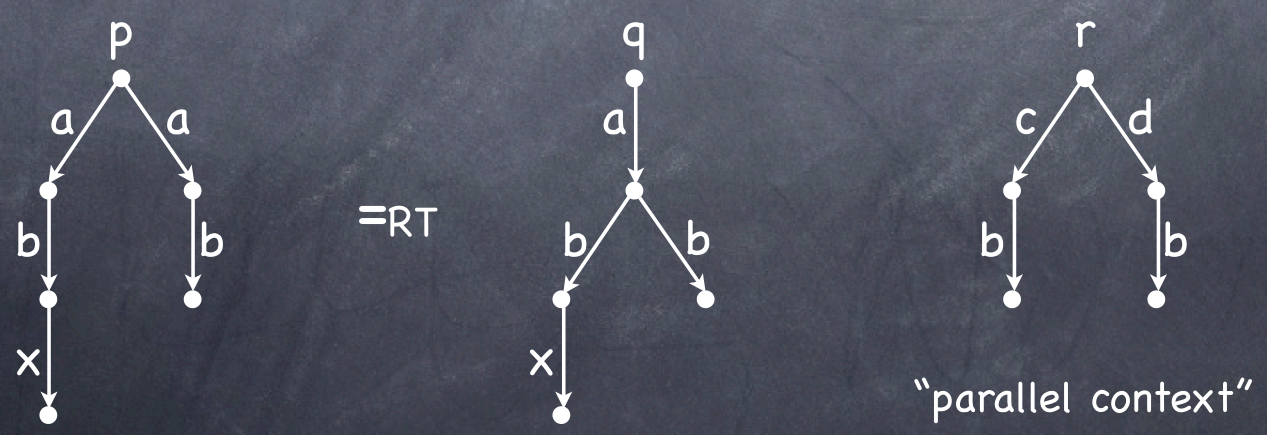
Full Abstraction wrt. Conjunction [FOSSACS'06]

- Ready-tree preorder:
  - $p \lesssim_{RT} q$ 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):
  - $p \lesssim_{F} q$ 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:
  - $\lesssim_{RT}$ is the largest precongruence wrt. $\wedge$ in $\lesssim_{F}$, i.e.,

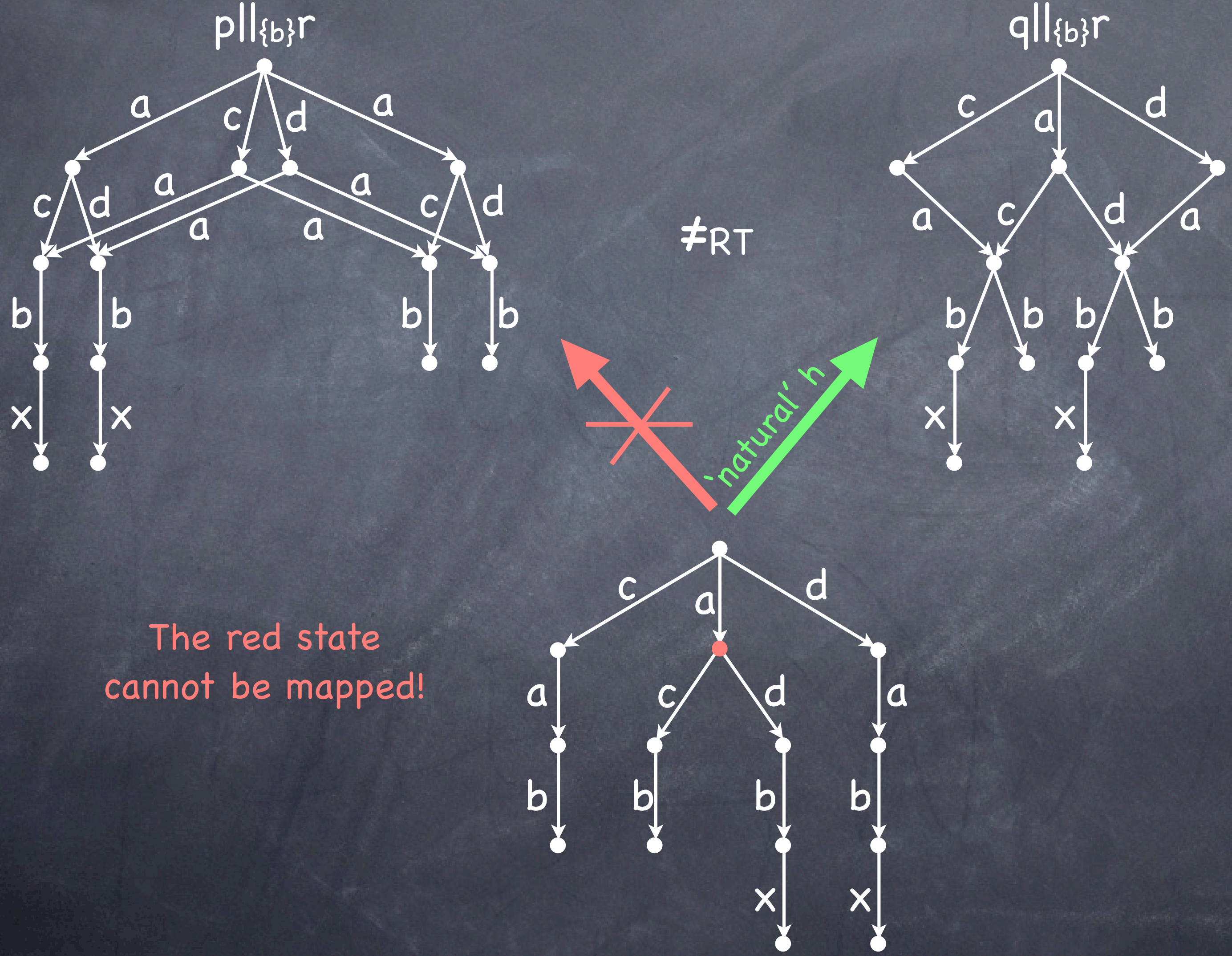
    $p \lesssim_{RT} q$ if and only if $\forall r.\ p \wedge r \lesssim_{F} q \wedge r$

Parallel Composition on Logic LTS

- Parallel composition $||_A$ as in CSP but with
  - τ's done first, in order to preserve τ-purity
  - $p||_A q$ inconsistent if p inconsistent <u>or</u> q inconsistent
- <u>Compositionality defect of the ready-tree preorder:</u>



Compositionality Defect Illustrated



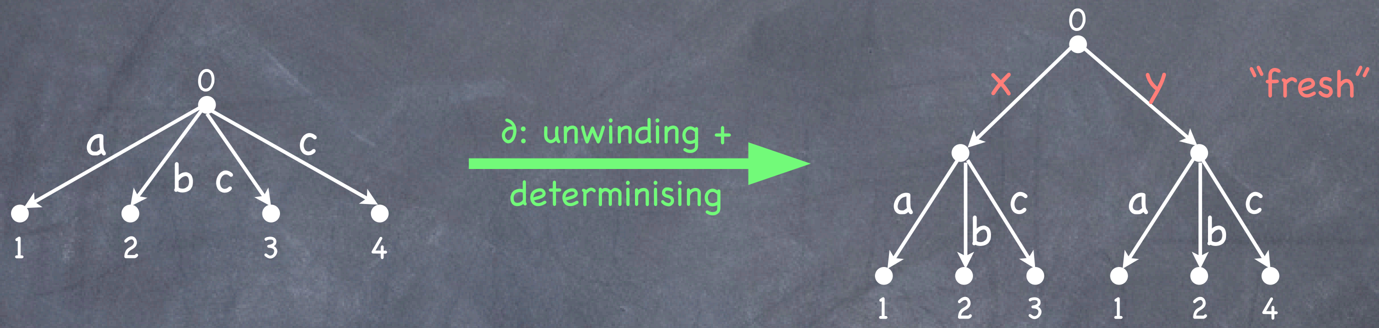
The red state
cannot be mapped!

Ready Simulation & Full Abstraction

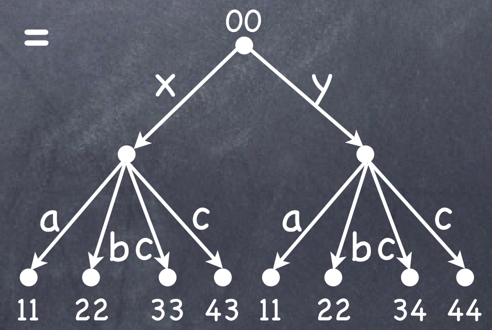
- Adaptation of ready simulation [Bloom/Istrail/Meyer, 1995] to Logic LTS, i.e., p $\leq_{RS}$ q if
  - Consistent steps of p can be matched by consistent steps of q
  - Stable states of p are matched by stable states of q that offer the same ready set
- Full-abstraction result:
  - $\leq_{RS}$ is the 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 ...

Some Insight Into the Full-Abstraction Proof

- Encode a process p's full behaviour by a ready tree ∂(p)



- La.: ∂(p) is a ready tree of $p\|_{\{a,b,c\}}\partial(p)$ =



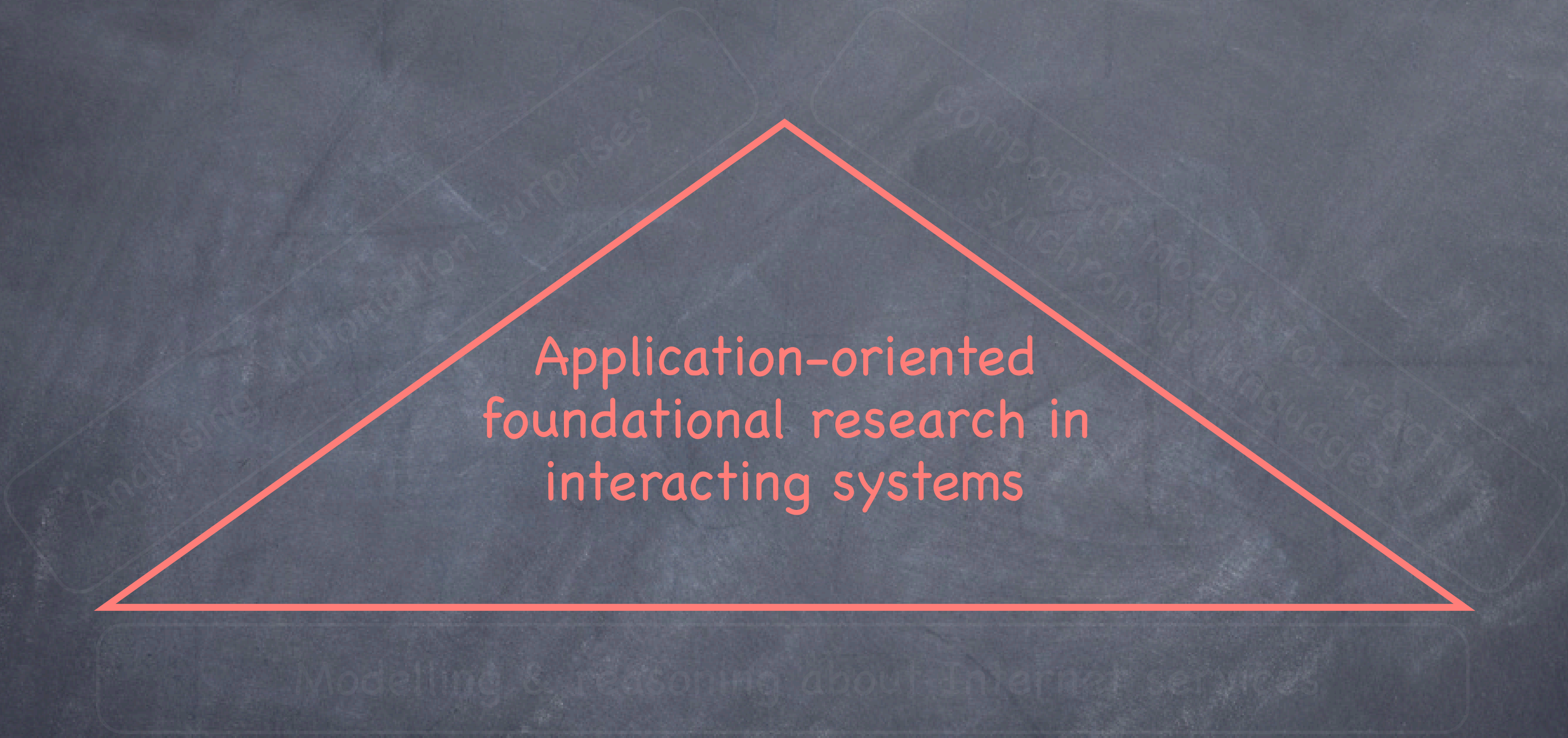
Logic Properties of Ready Simulation

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

Conclusions & Current/Future Work

- Ready simulation is "logical"!
  - Logic LTS is suitable for modelling and reasoning about specifications given in a mixed operational and logic style
- Extensions:
  - Adding process-algebraic operators, e.g., external choice ($p \Box q$) and CSP-style hiding (p/h)
  - Adding logic operators, e.g., disjunction ($p \vee q$ - internal choice) or release ($p \mathrel{R} q$ - temporal safety property)
  - Adding recursion operators ($\mu x.p$, $\nu x.p$)
  - Investigating axiomatisations

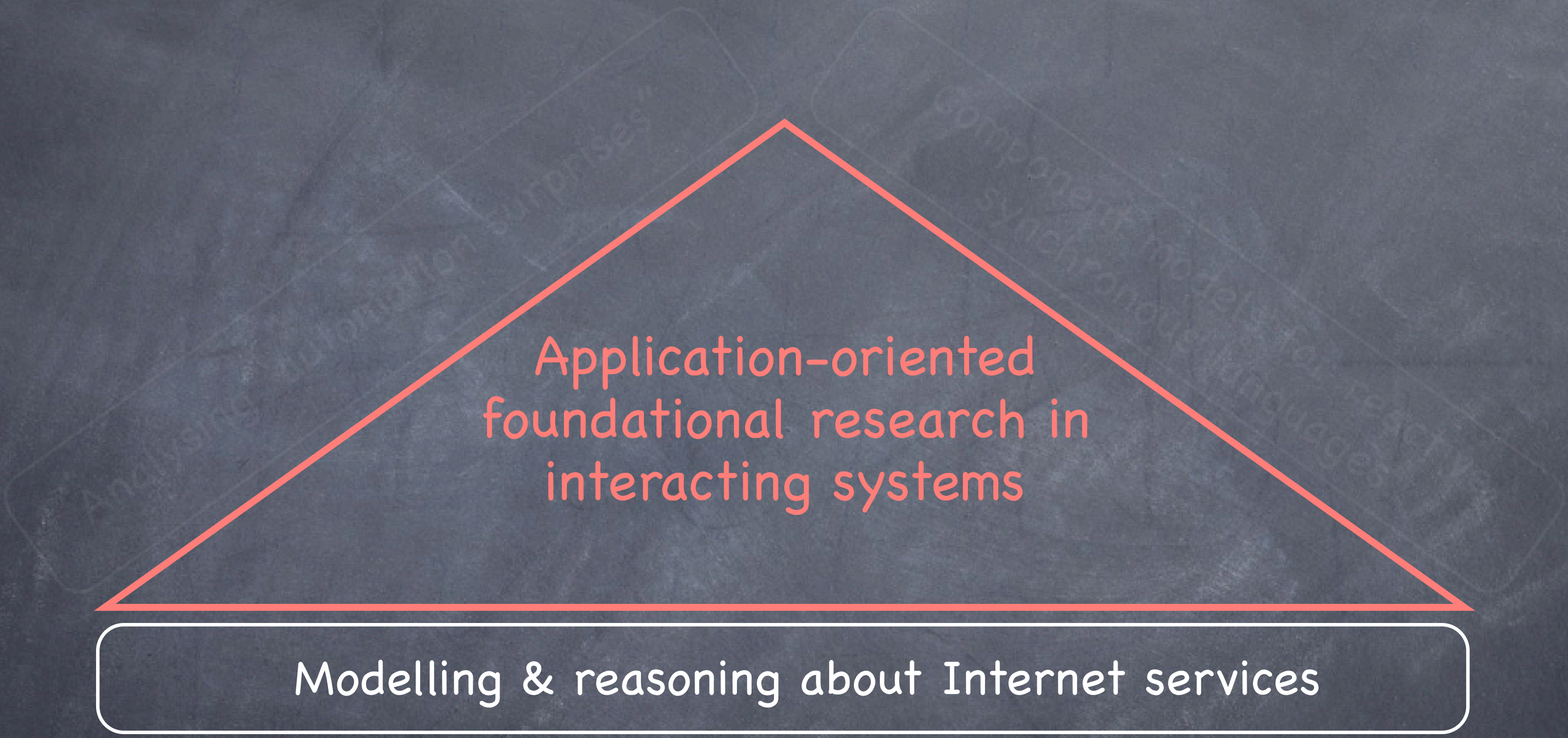
Desert: Potential Synergies with SICSA Themes



My research lies within Theme 3:

- Focuses on abstract modelling and reasoning
- Advocates compositional methods for dealing with complexity

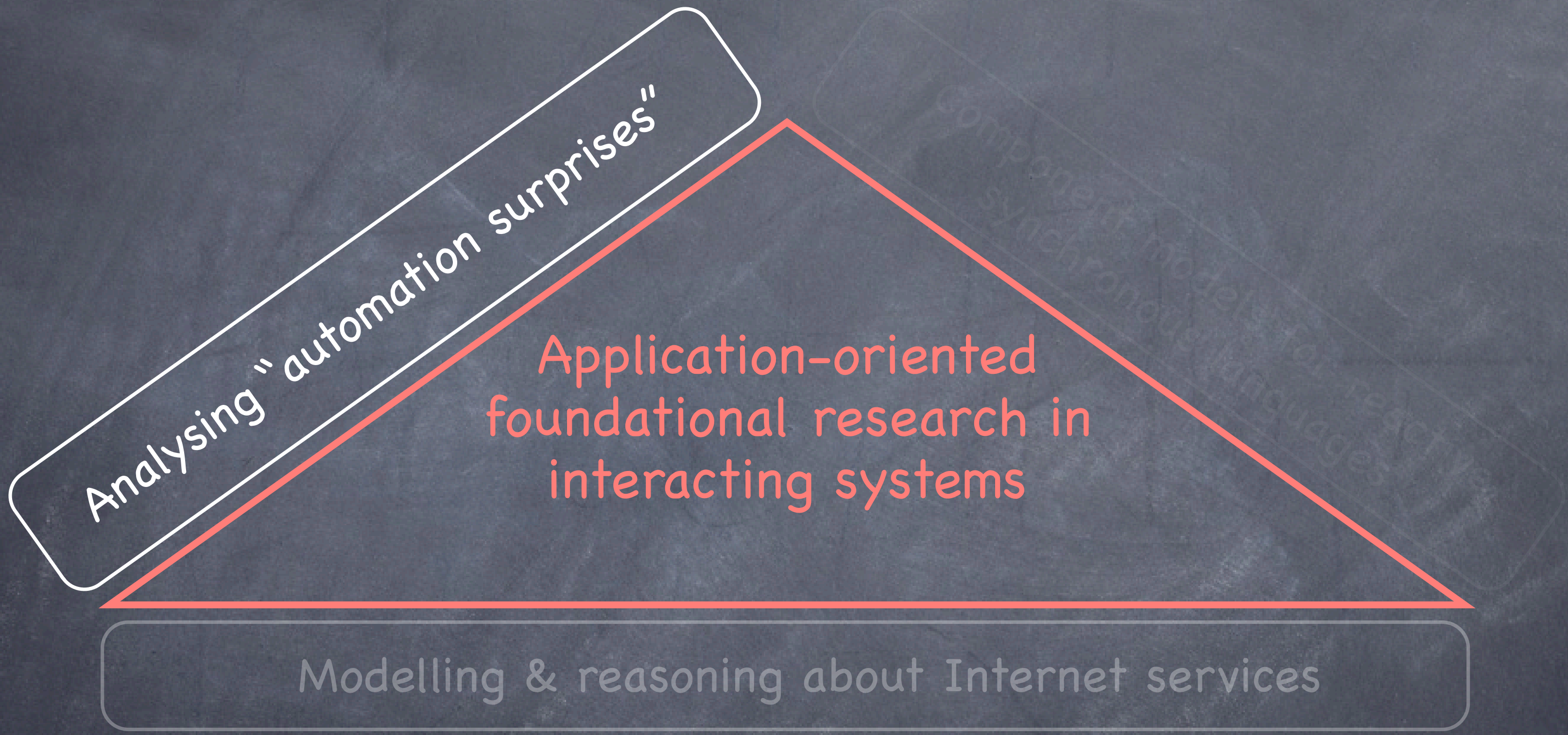
Potential Synergies with SICSA Themes



Synergy with Theme 1 – Internet Services:

- Application of my work on multi-clock process algebra to the orchestration of peer-to-peer web services; challenge lies in extending this work to mobility and data

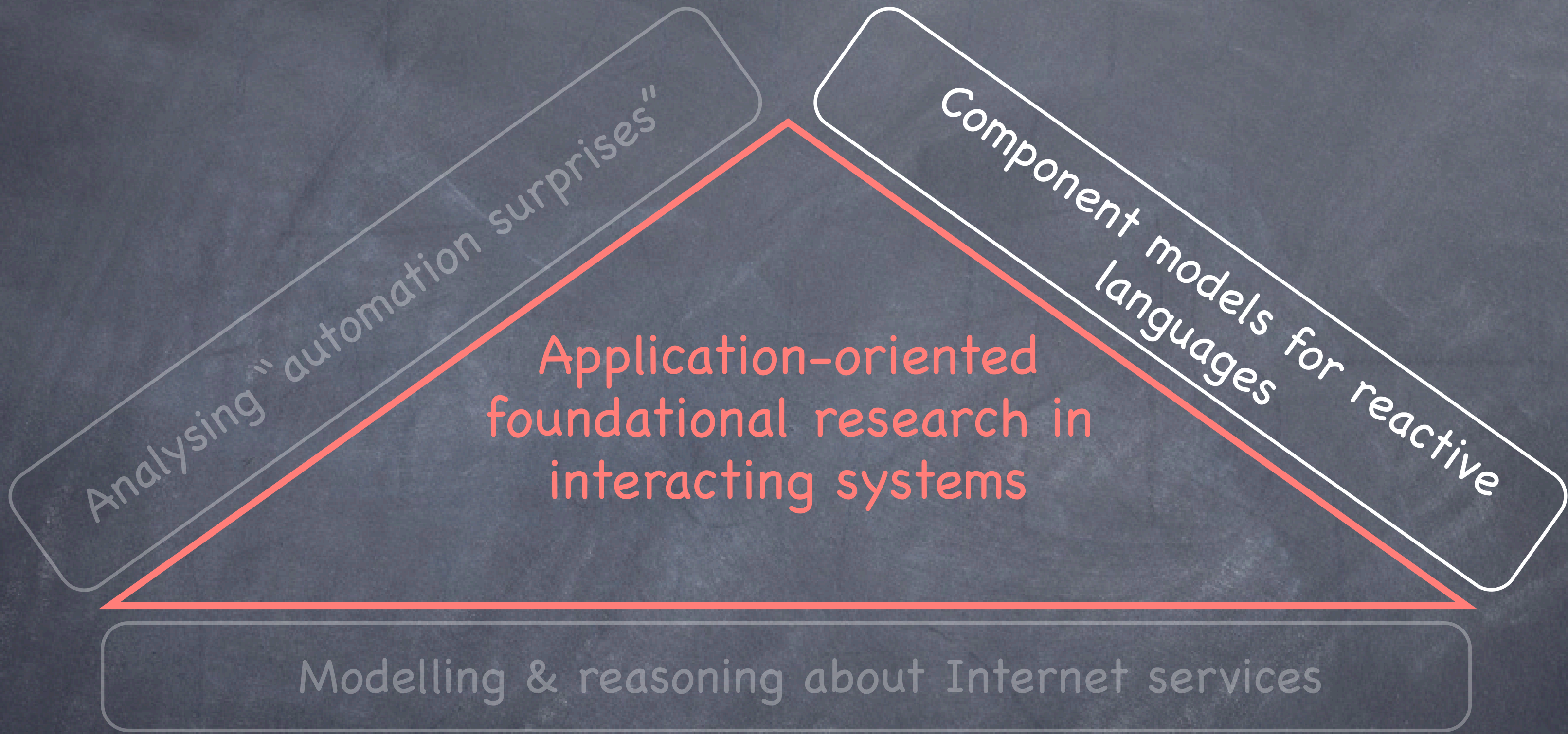
Potential Synergies with SICSA Themes



Synergy with Theme 2 - Human Information Interface:

- Previous experience gained at NASA/ICASE with applying model checking to analysing sources of mode confusion related to cockpit automation

Potential Synergies with SICSA Themes



Synergy with Theme 4 - Systems of Systems:

- Interoperability of reactive languages, based on reactive types
- Co-ordination of synchronous reactive components via an asynchronous communications layer (GALS)

Thank You for Listening!

Analysing "automation surprises"

Component models for reactive languages

Application-oriented foundational research in interacting systems

Modelling & reasoning about Internet services

Questions?