USING PARTIAL MODEL SYNTHESIS TO SUPPORT MODEL INTEGRATION IN LARGE-SCALE SOFTWARE DEVELOPMENT

Marsha Chechik and Rick Salay

GEMOC'2013

September 2013

Motivation

- Global software engineering projects
  - Highly distributed
  - Involve many stakeholders which produce heterogeneous models reflecting different concerns
- GEMOC initiative:
  - Find ways to support collaboration and interoperability at the integration points between models
- Key challenge:
  - Management of dependencies between models!

GEMOC Approach

Clearly define interface specifications early in the development lifecycle:

- E.g., a set of operation signatures or detailed interaction scenarios, properties, invariants

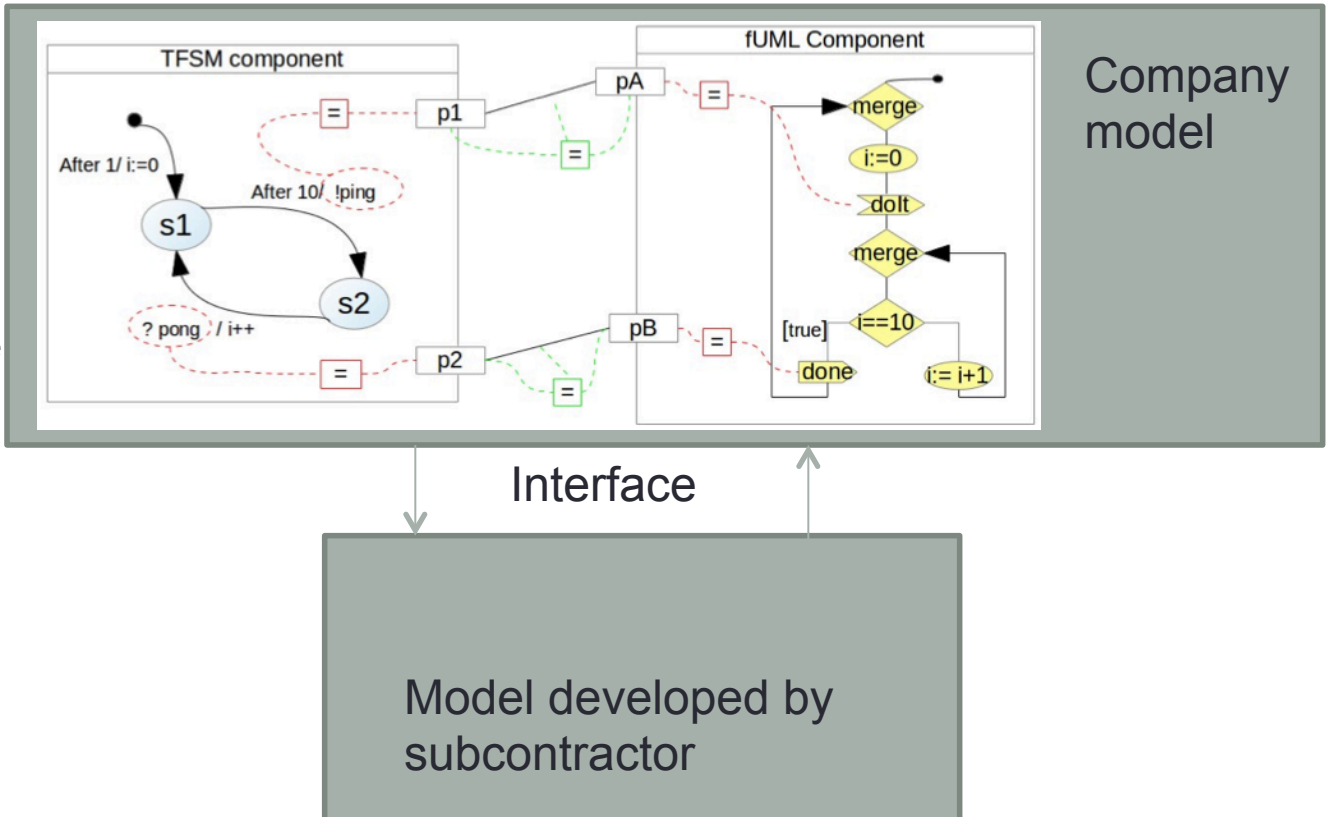
☑ These are useful for guiding model development

✖ Not well suited for dynamic techniques such as testing, simulation, model-checking

Our proposal

Automatic synthesis of partial models from interface specifications that act as model stubs and can be used as temporary stand-ins for the models to allow simulation and other analysis.

Partial model: a set of possible models admissible by the specification

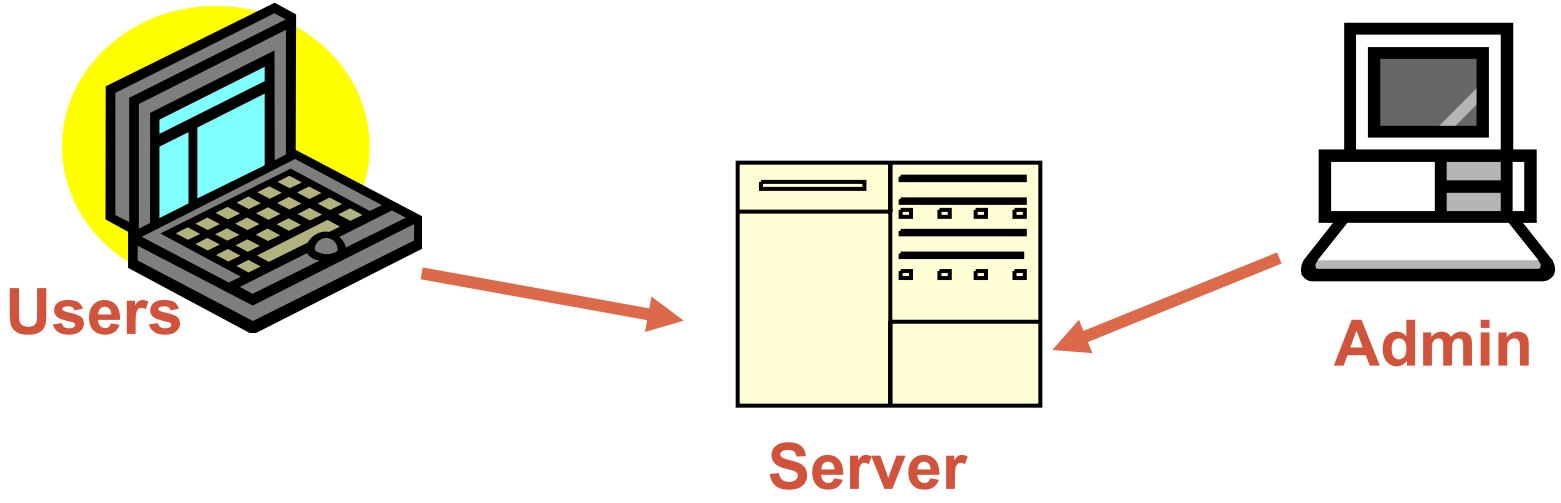


Outline

- Motivation
- Illustration: Web-mail system
- Classical approach:
  - Synthesis from Scenarios
  - Synthesis from Global Properties
- Partial modeling approach:
  - Synthesis from Scenarios
  - Synthesis from Global Properties
  - Merge
- Status
- Connection to GEMOC

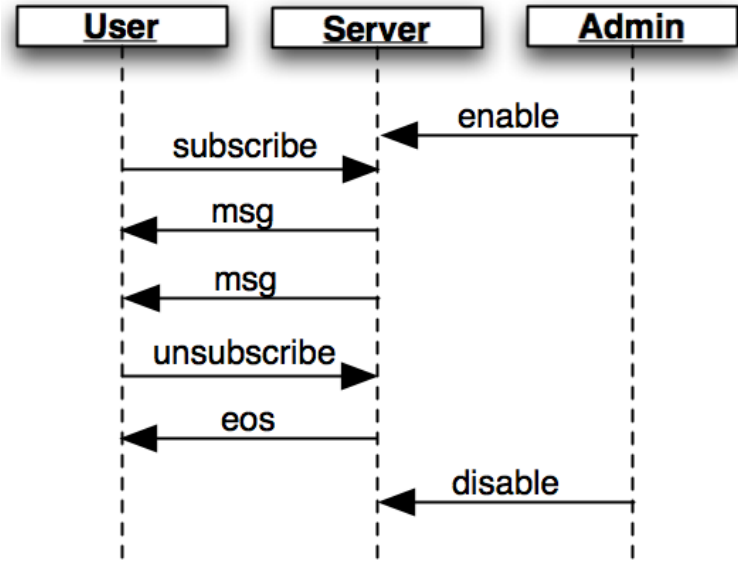
Web-mail System

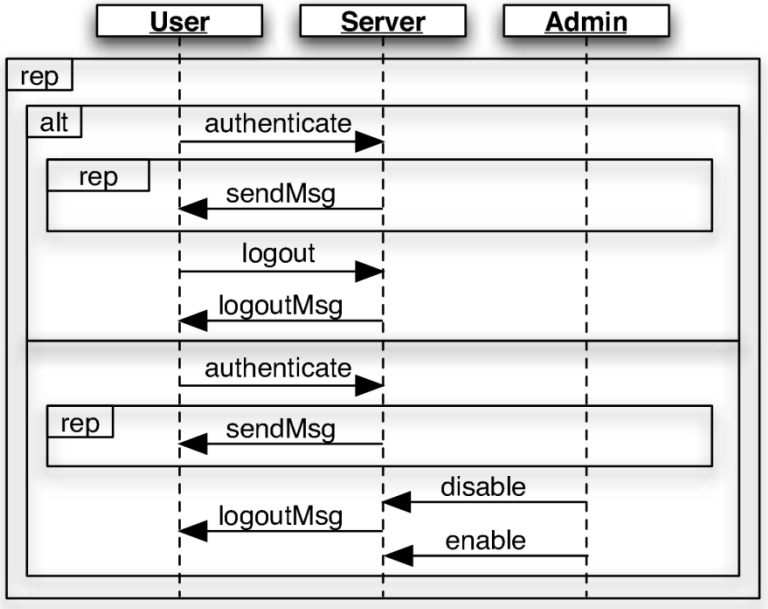
A toy example to illustrate our approach…



Description consists of a set of scenarios and FLTL properties

Web-mail Scenarios





Web-mail Properties

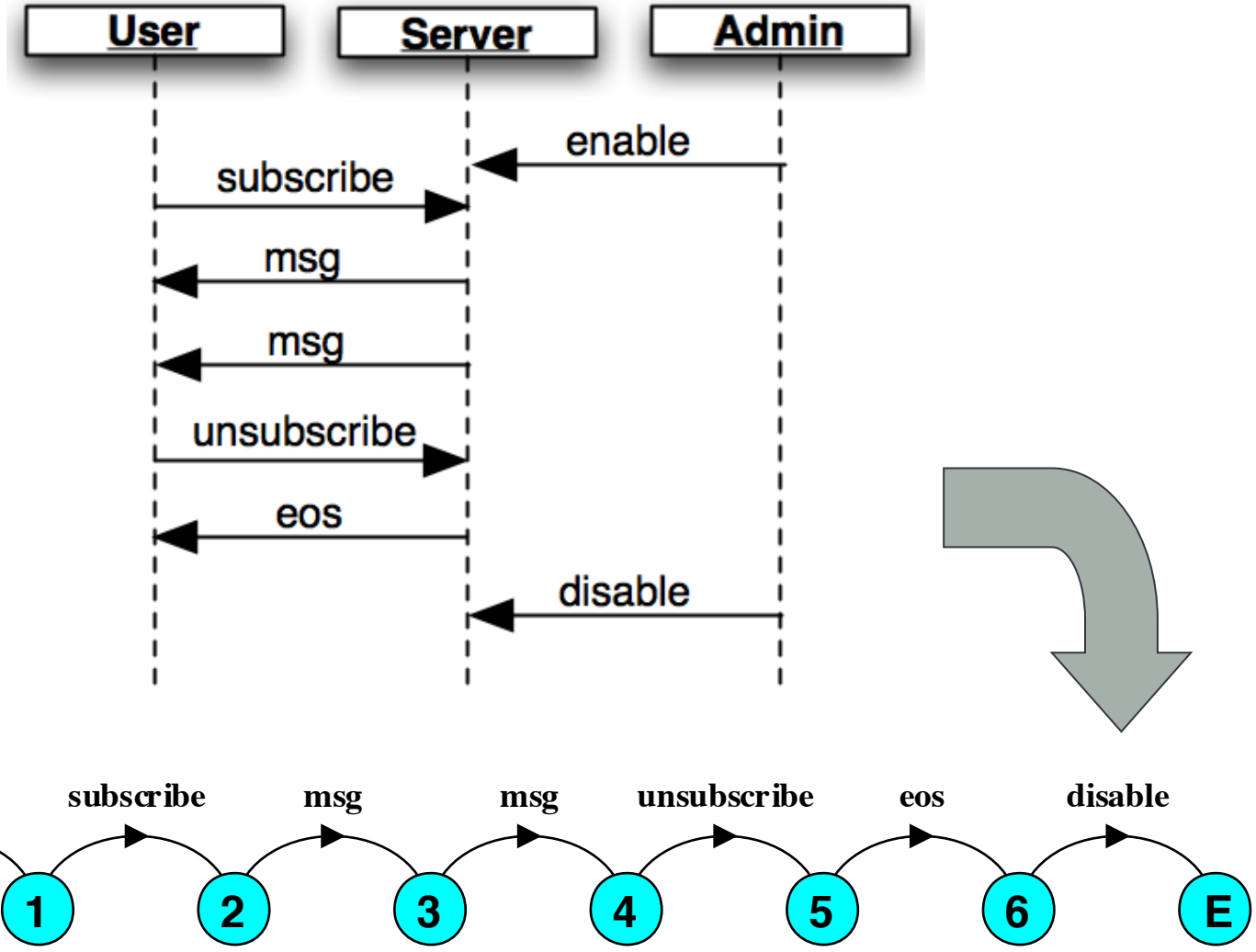
Expressed in FLTL

fluent Registered = <enable, disable> initially TRUE
fluent LoggedIn = <authenticate, {logout, disable}> initially FALSE

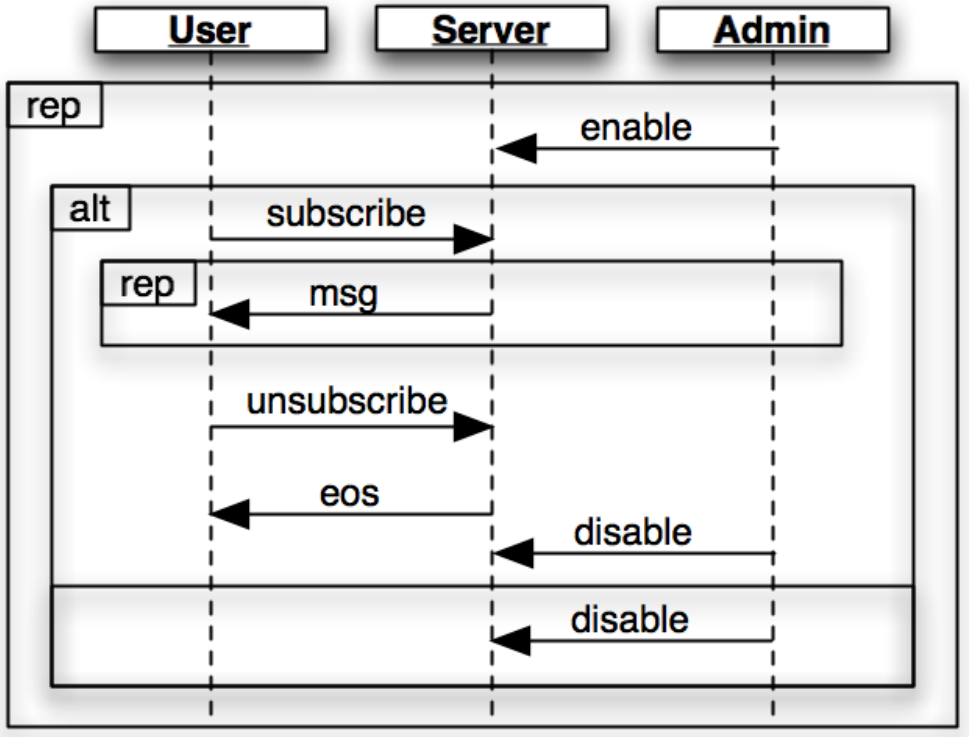
Properties

LegalAccess:	[](LoggedIn -> Registered)
PrivateAccess:	[](sendMsg ->LoggedIn)
Logouts are ack'd:	[](logout -> X logoutMsg)

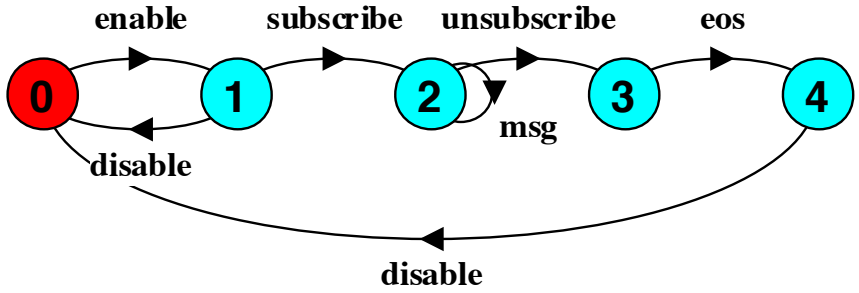
(classical) Synthesis from Scenarios



Synthesis from Complex Scenario Specifications

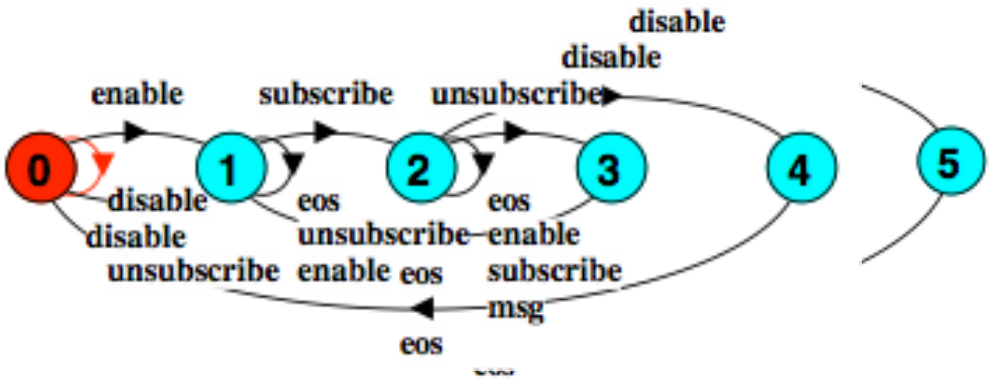


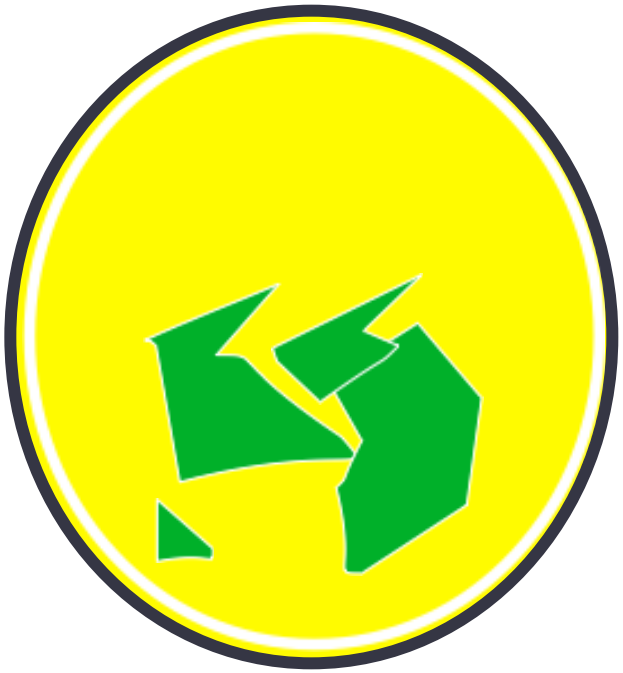
- Support for repetitions, alternatives, references, triggers, symbolic instances, preemption, broadcasting...



Scenarios are Partial Specifications

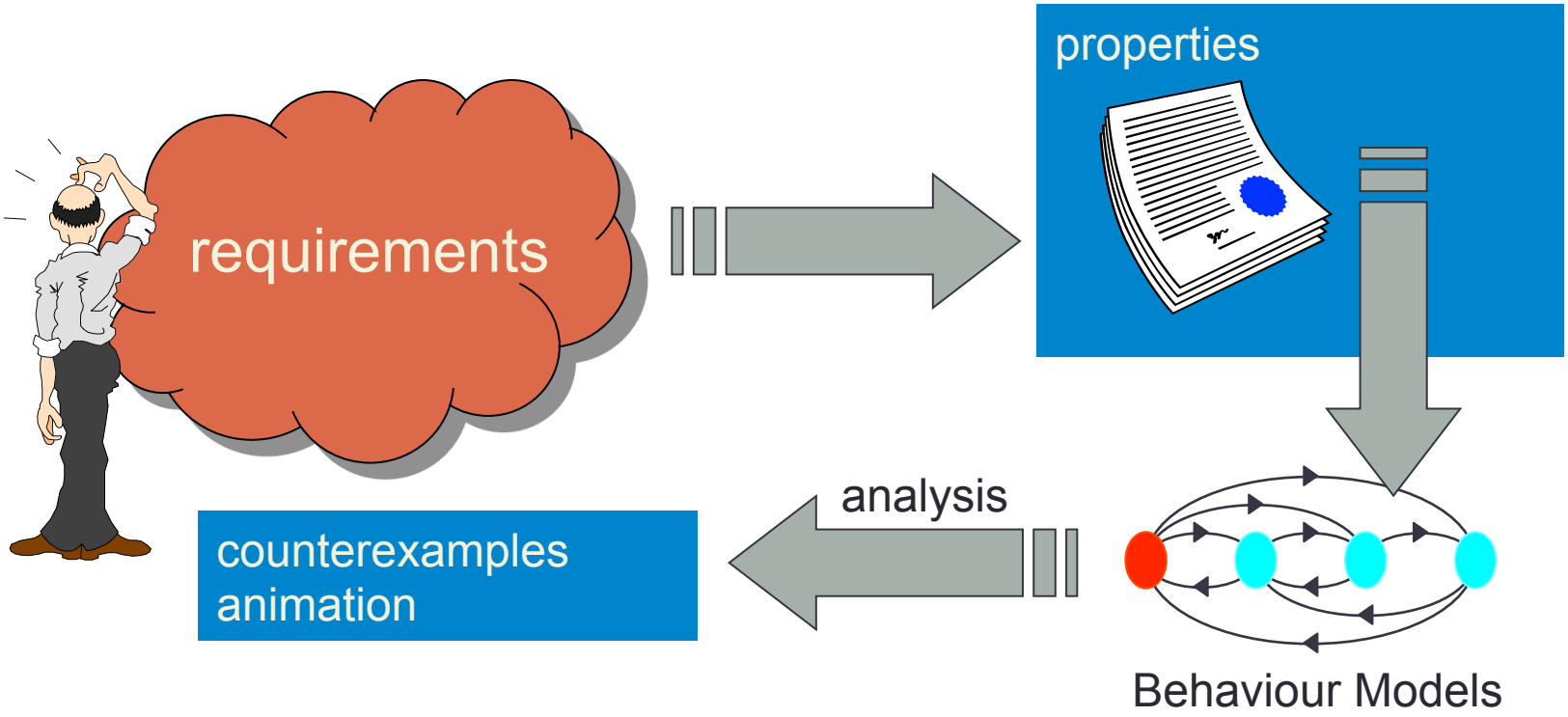
- A model synthesised from scenarios
  - Describes behaviour that the system must be capable of exhibiting
  - Does not describe all the required behaviour
- Elaboration = “adding behaviour”
  - i.e., preserving trace inclusion or simulation





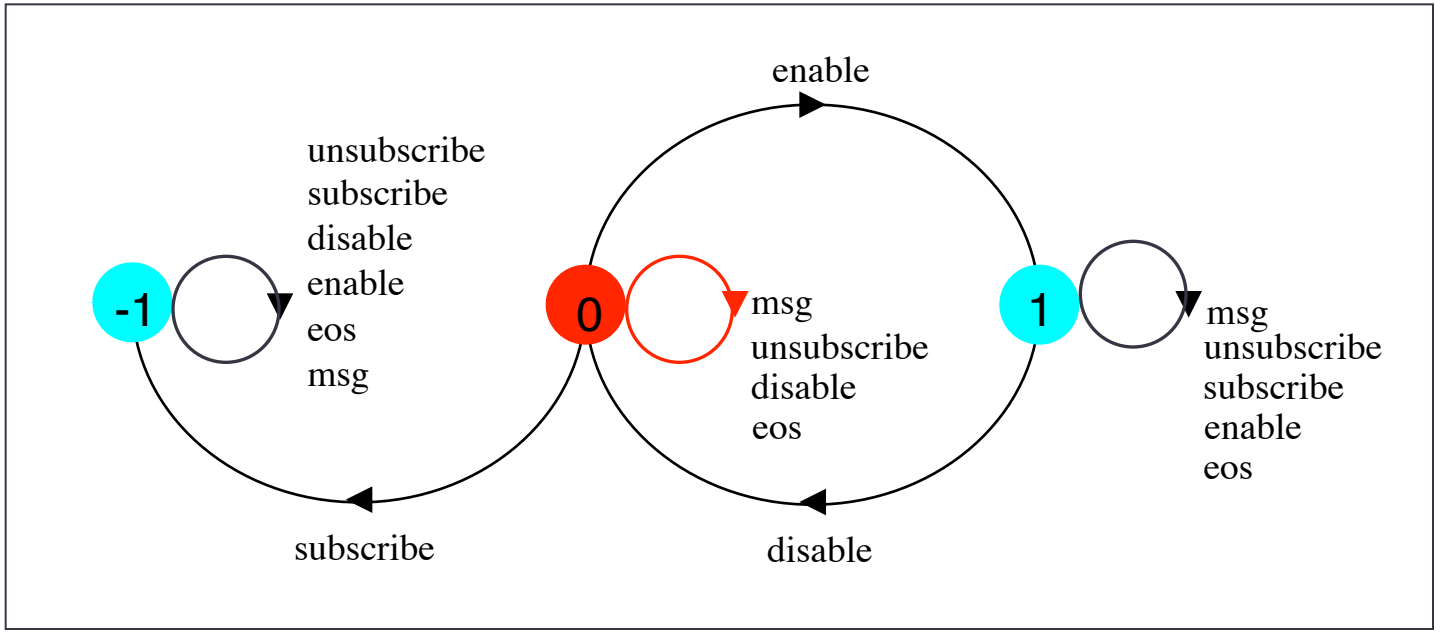
Synthesis from Declarative Specifications

- Build models automatically from assertions about the intended system behaviour.



FLTL Safety Properties to LTS

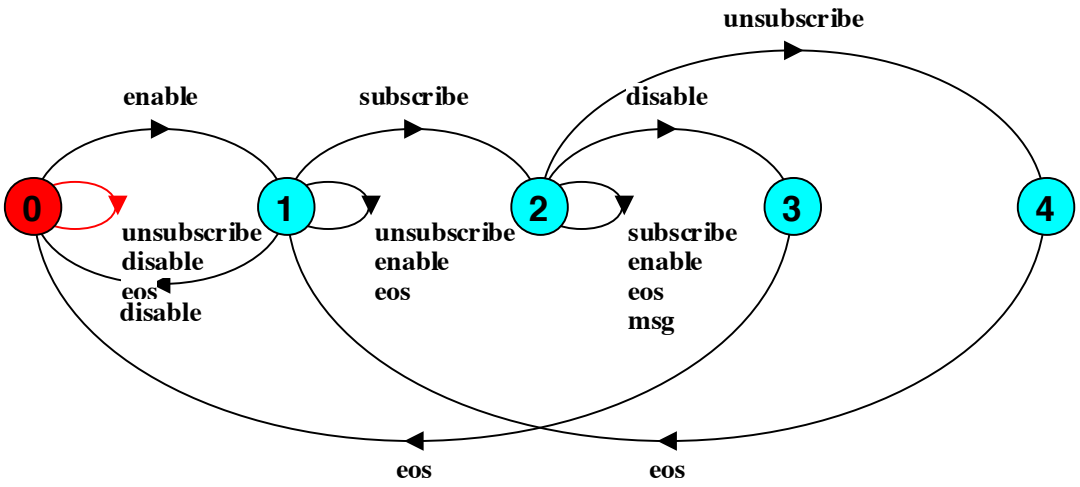
1. Apply FLTL to Büchi construction [Gianna'03]
2. Remove accepting state and all transitions leading to it.



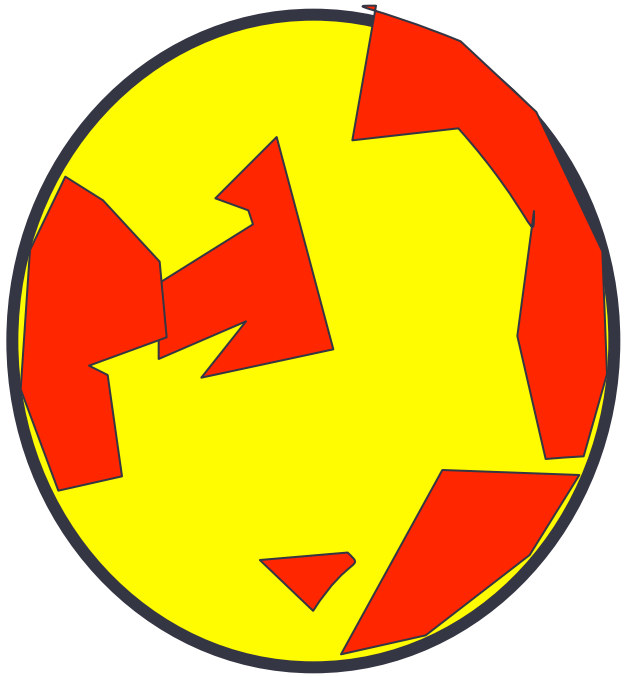
constraint LegalAccess = [](Subscribed -> Enabled)

Models Synthesized from Properties are Partial too!

- A model synthesized from a property P
  - describes all behaviour that does not violate P, or alternatively
  - describes behaviour that the system may exhibit
- Not all described traces are required to be provided by the system
- Elaboration = “removing behaviour”
  - i.e., preserving trace inclusion or simulation

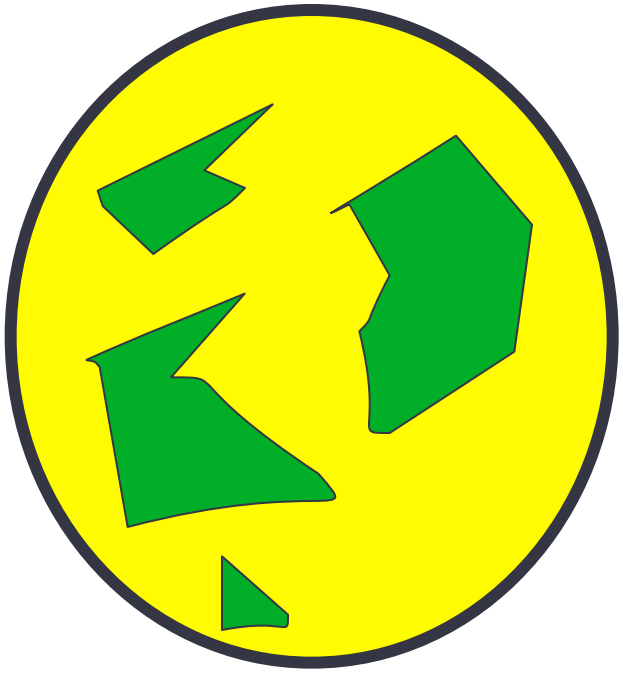


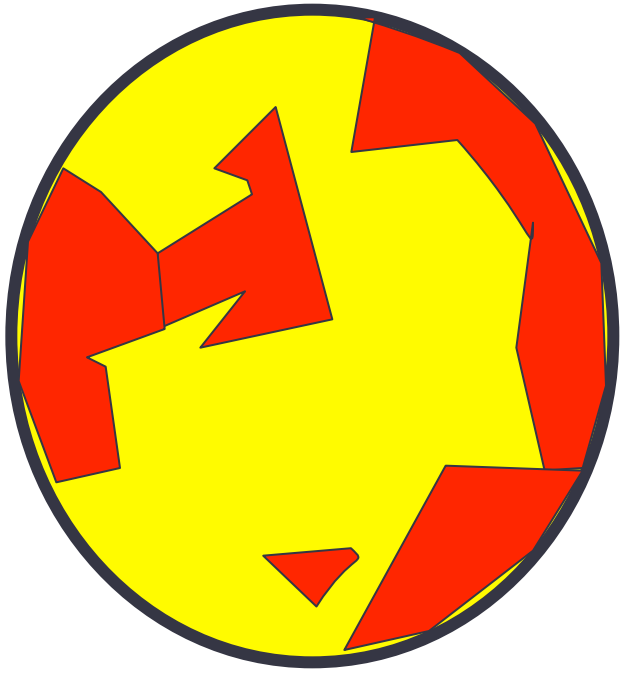
PrivateAccess



Scenarios and Properties are Complementary

- Scenarios...
  - describe traces we know we want.
  - provide a lower bound to intended system behaviour.
- Properties...
  - describe what we know we don't want.
  - provide an upper bound to intended system behaviour

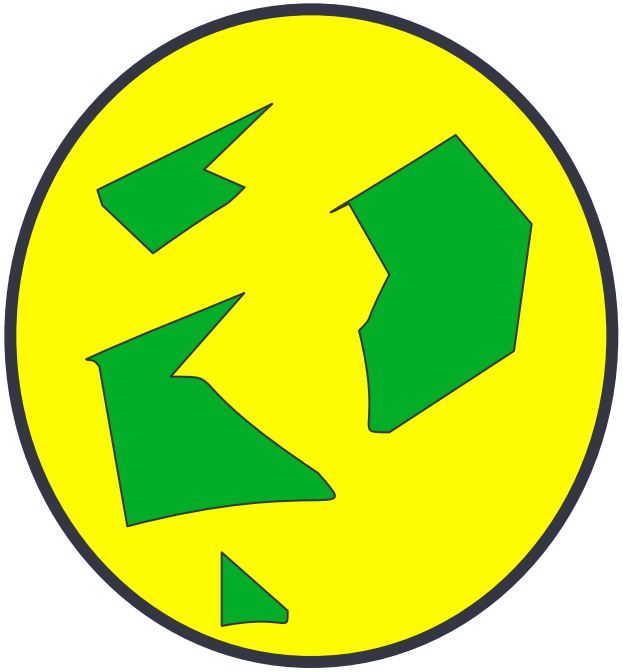




LTS are Not Expressive Enough...

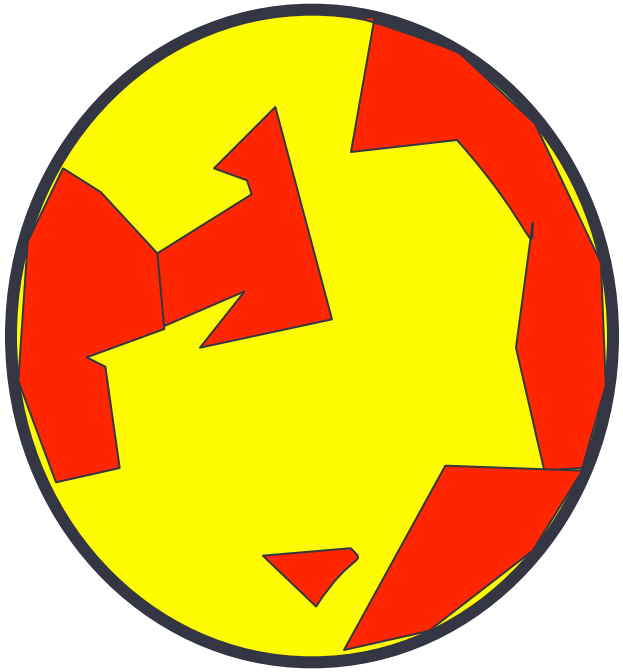
- 2 valued behaviour models cannot capture the “middle ground”

LTS from Scenarios...



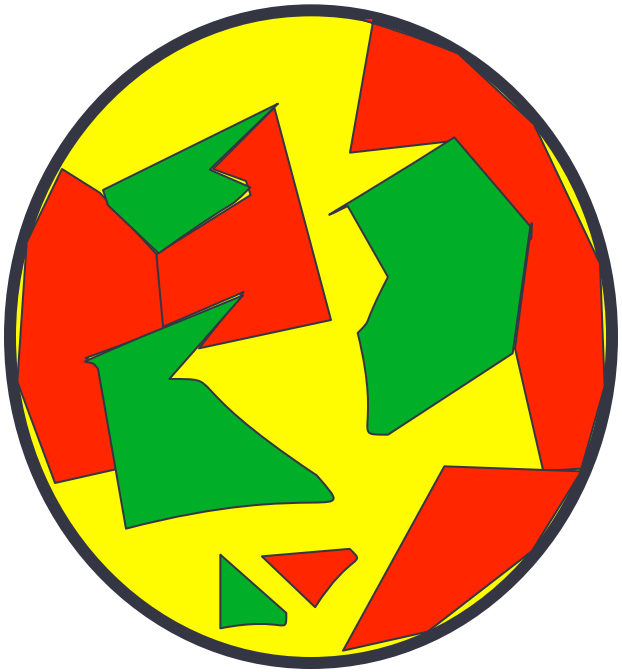
Green = LTS traces
Yellow = everything else

LTS from Properties...



Yellow = LTS traces
Red = everything else

LTS from Scenarios and Properties...



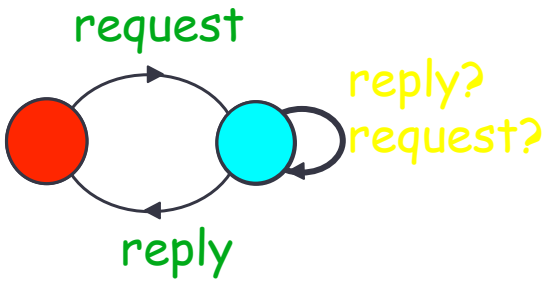
Yellow = ? Red = ?
Green = ?

Outline

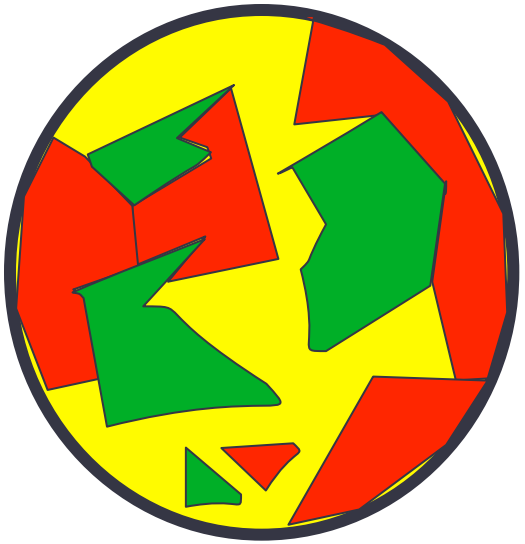
- Motivation
- Illustration: Web-mail system
- Classical approach:
  - Synthesis from Scenarios
  - Synthesis from Global Properties
- Partial modeling approach:
  - Synthesis from Scenarios
  - Synthesis from Global Properties
  - Merge
- Status
- Connection to GEMOC

Modal Transition Systems (MTS)

- Larsen et. al. 1988 -

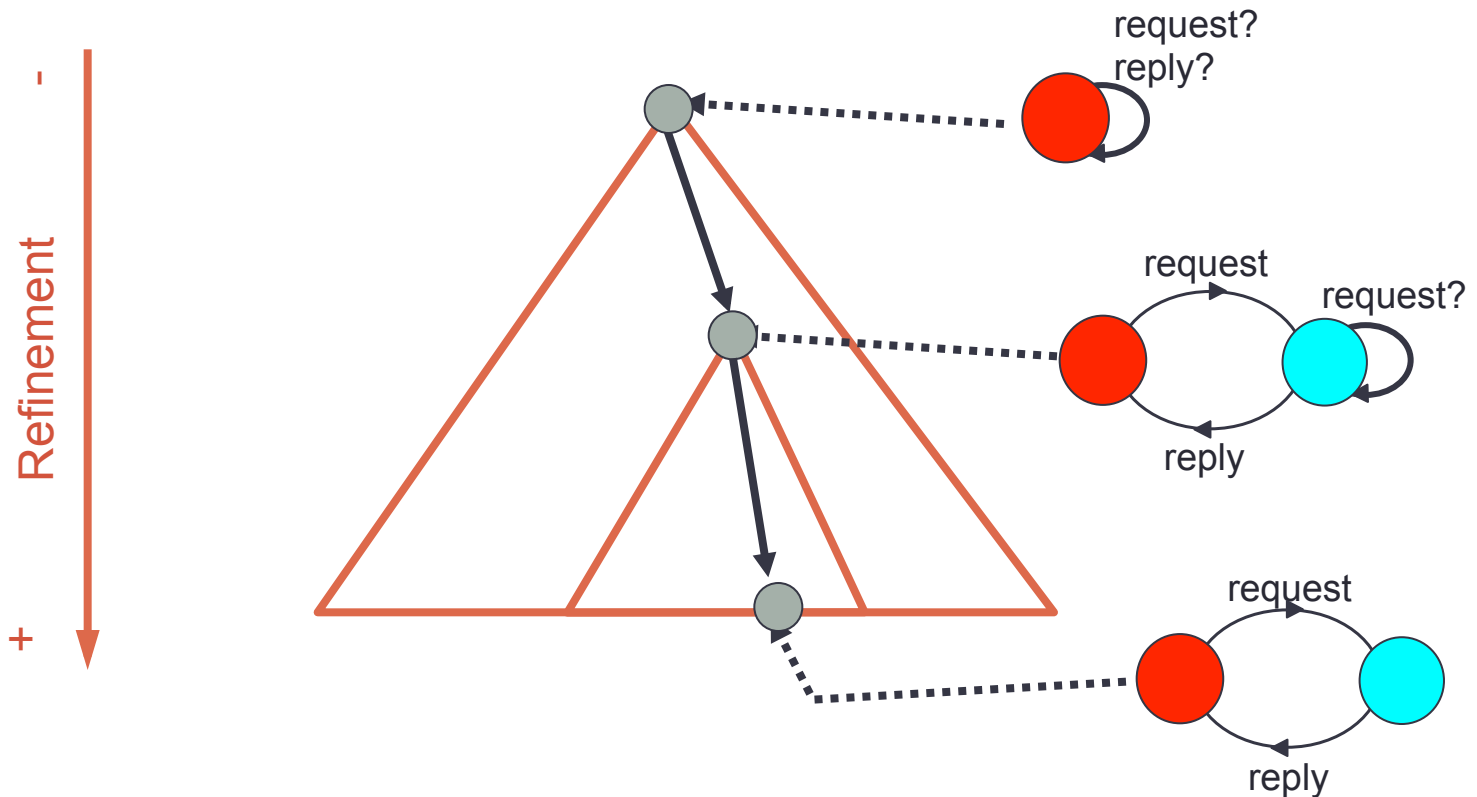


- Extension of LTS:
  - Required transitions
  - Possible transitions
- They describe
  - Explicitly what we do not know
  - Implicitly what we prohibit

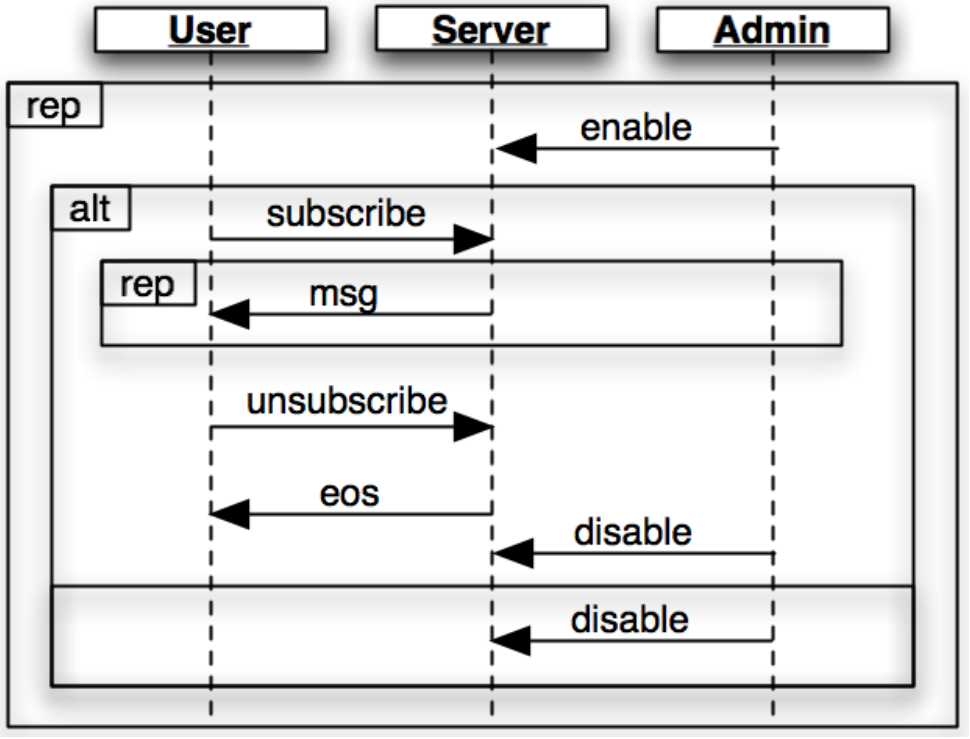


MTS Refinement

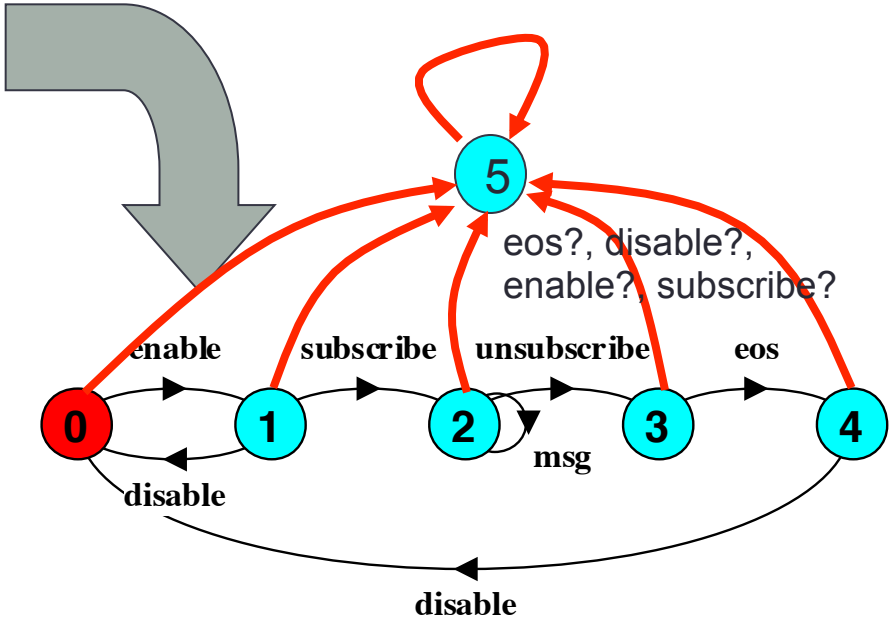
- Intuitive semantics: “more defined than”, “has more information than”
- Refinement is inclusion of implementations



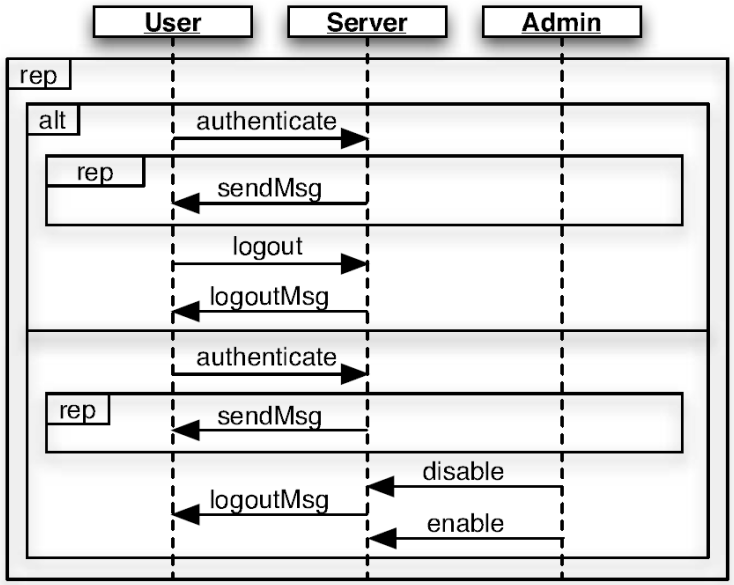
Synthesis of MTS from Scenarios

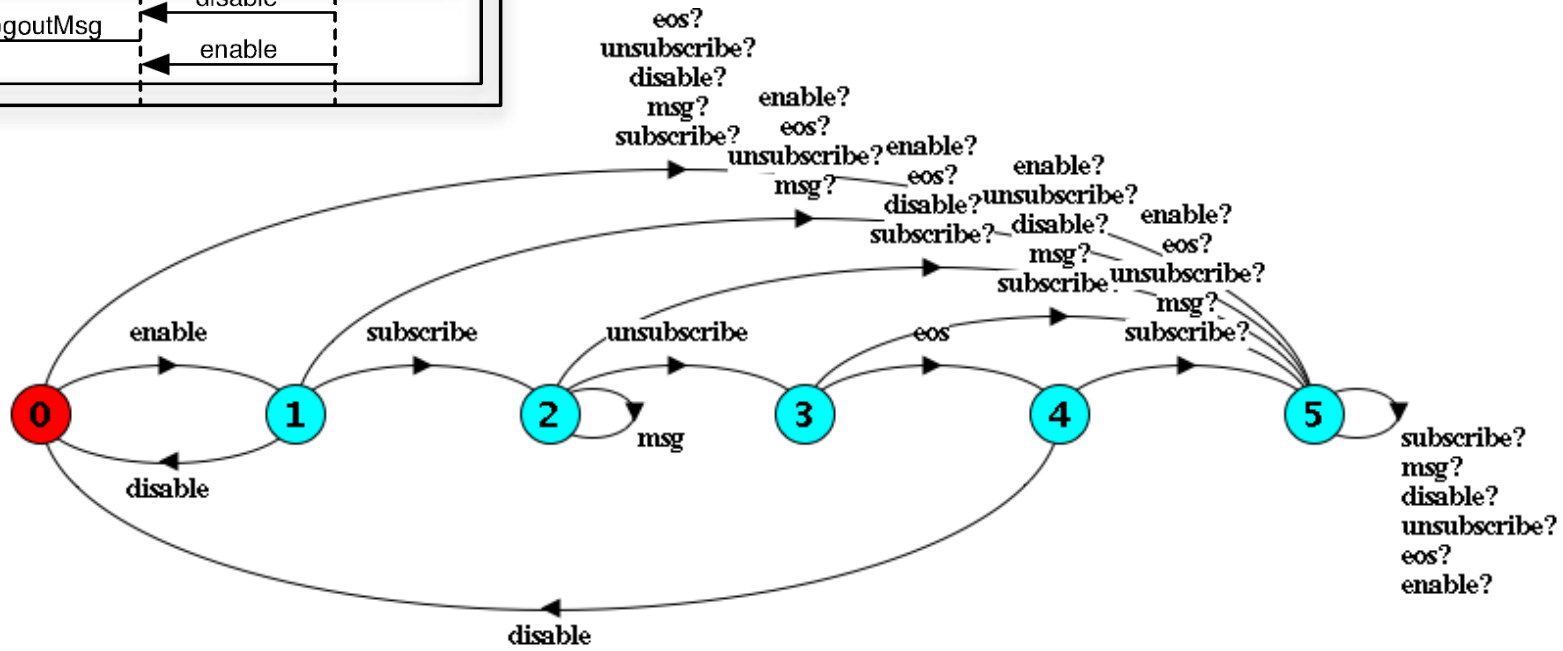


1. Synthesize LTS as usual
2. Add sink state
3. For all state s and label e, if e is not enabled on s, add a may-transition (s, e, sink)



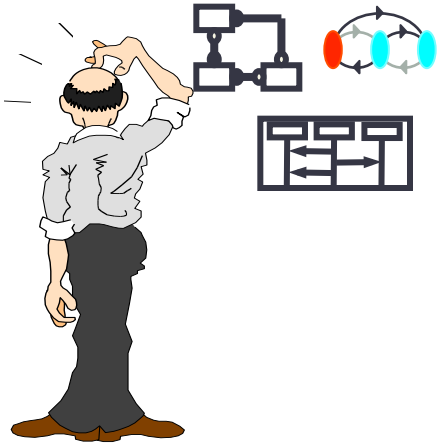
Synthesis of MTS from Complex Scenario



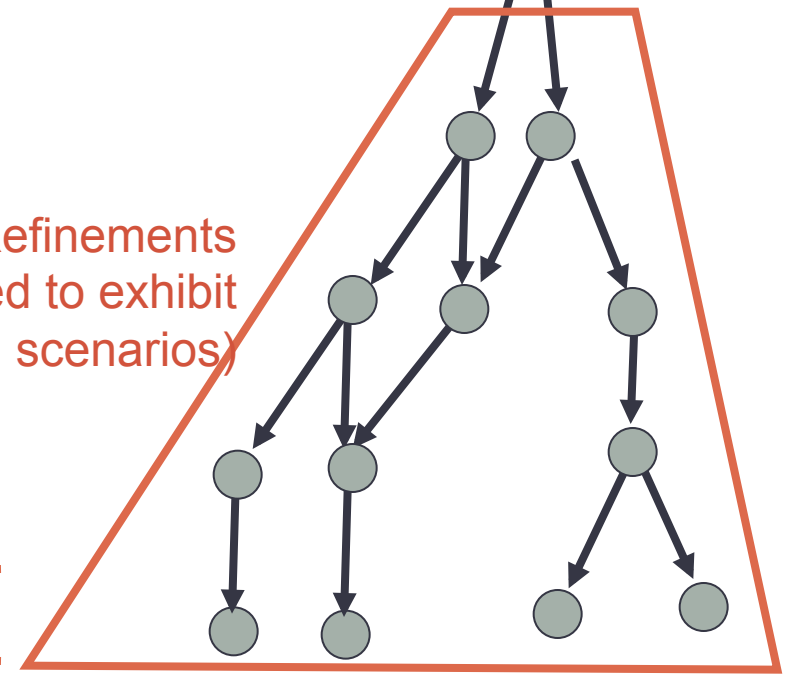


MTS Synthesis from Scenarios

- From architectural specifications
  - Structural descriptions + scenarios



Synthesized MTS



Refinements
(Guaranteed to exhibit
traces described in scenarios)

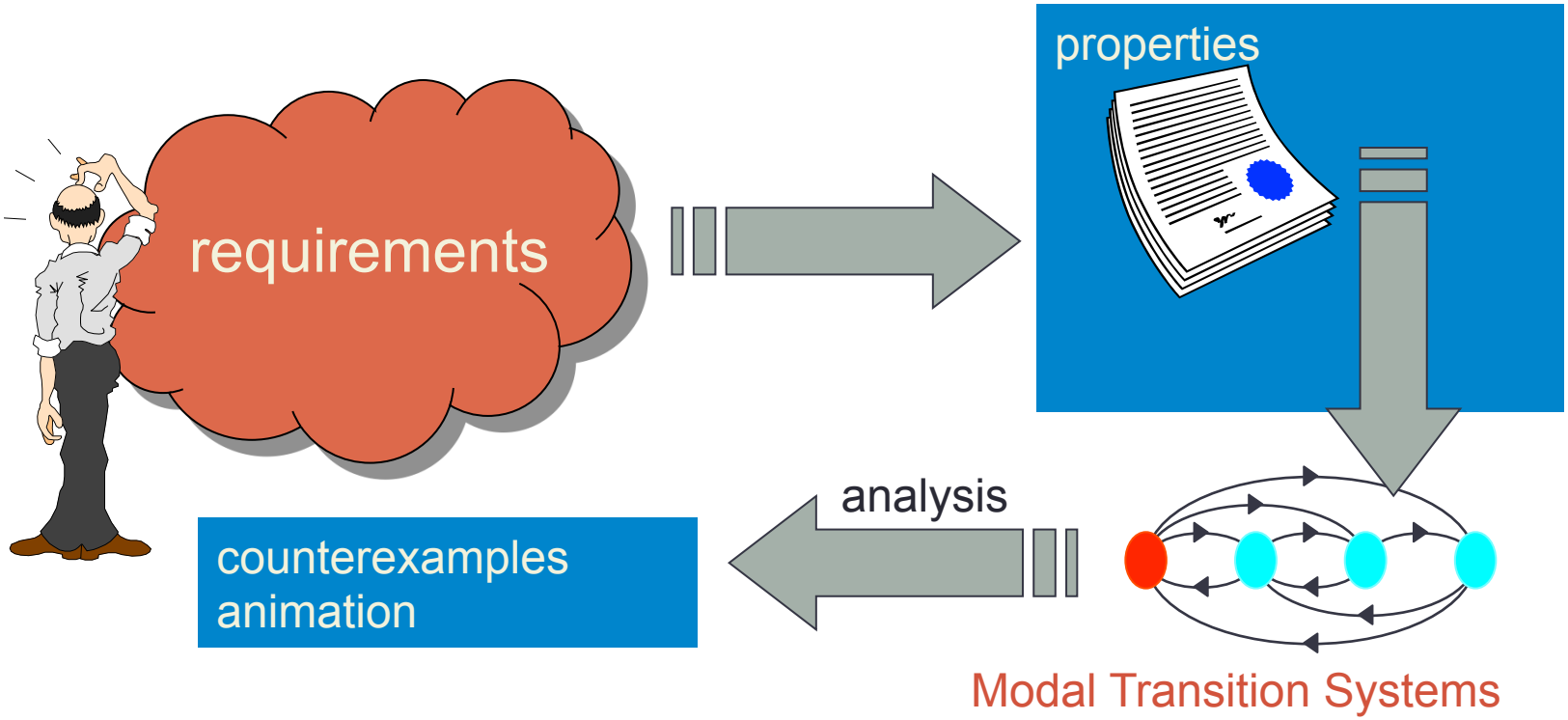
Implementations of
Scenario Specification

Note

- All of the above observations apply to synthesis from architectural descriptions, not just from scenarios

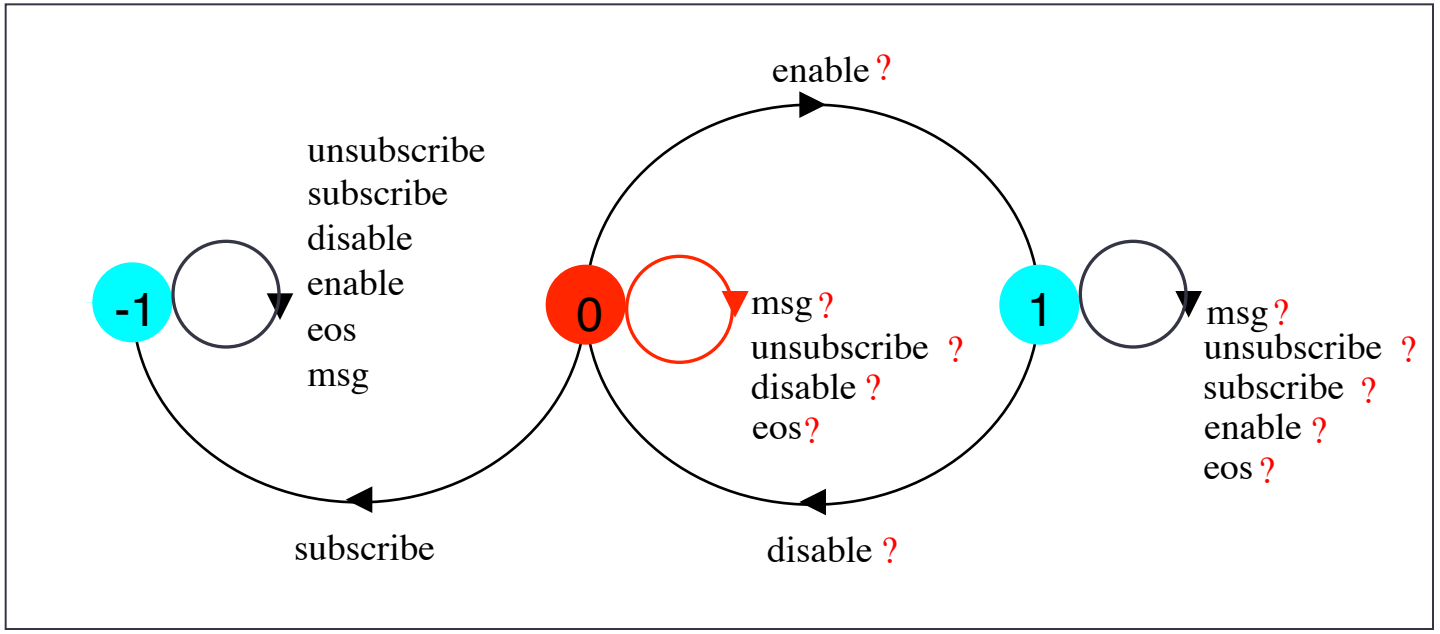
Synthesis of MTS from Declarative Specifications

- Build MTS models automatically from assertions about the intended system behaviour.



FLTL Safety Properties to MTS

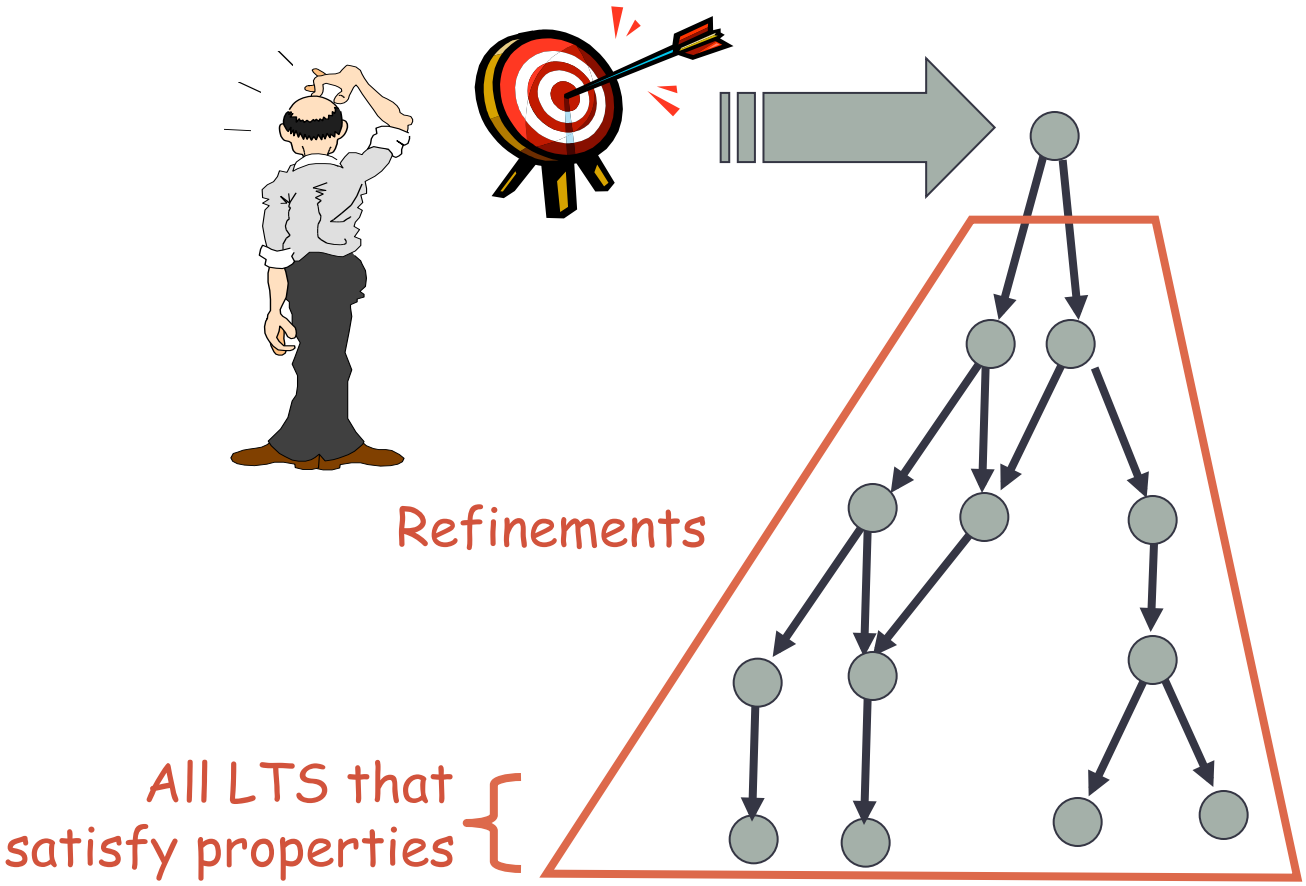
1. Apply FLTL to Büchi construction [Gianna'03]
2. Remove accepting state and all transitions leading to it.
3. If a state has more than one outgoing transition, make them all may-transitions



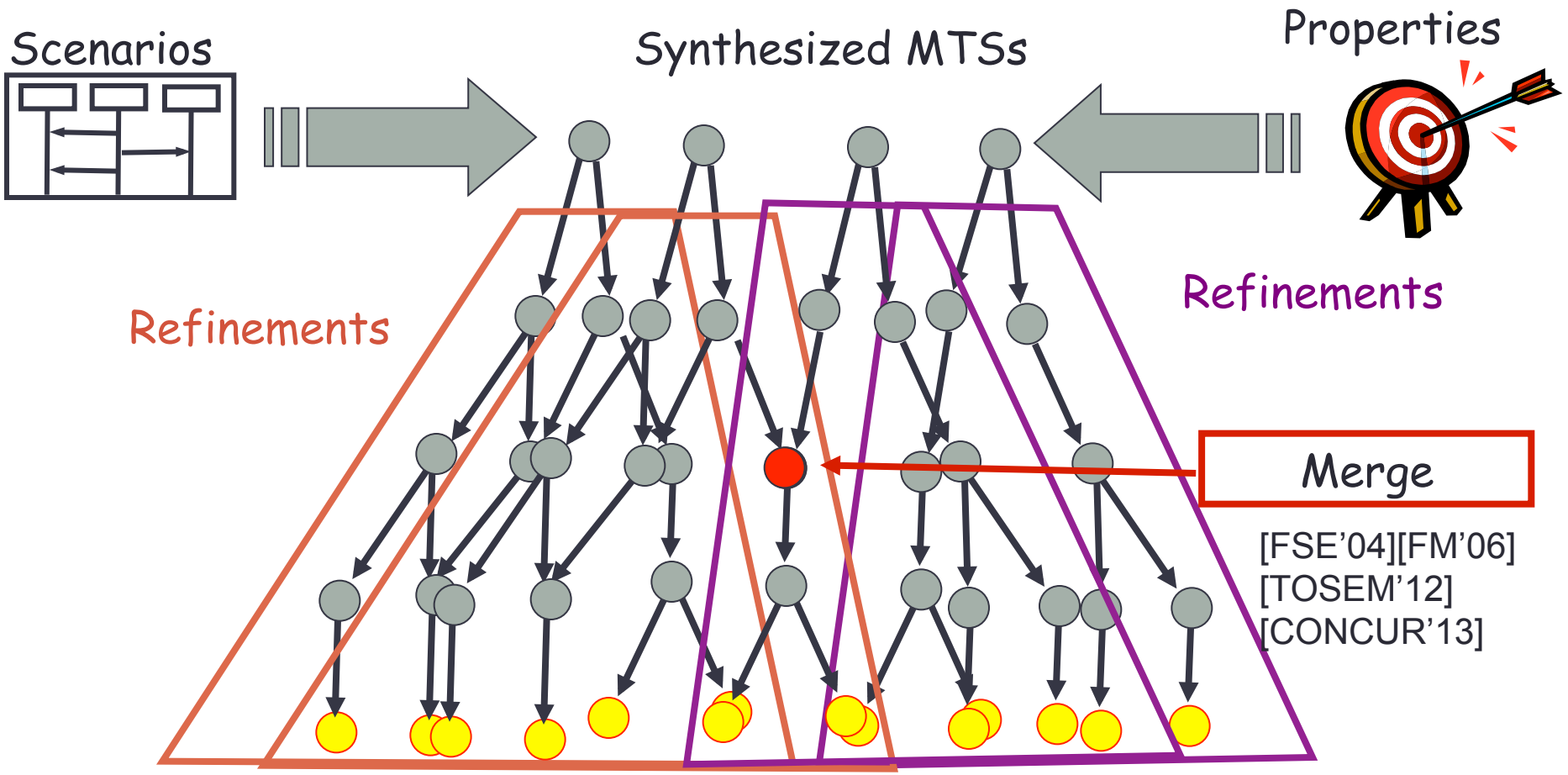
constraint LegalAccess = [](Subscribed -> Enabled)

Synthesis from Safety Properties

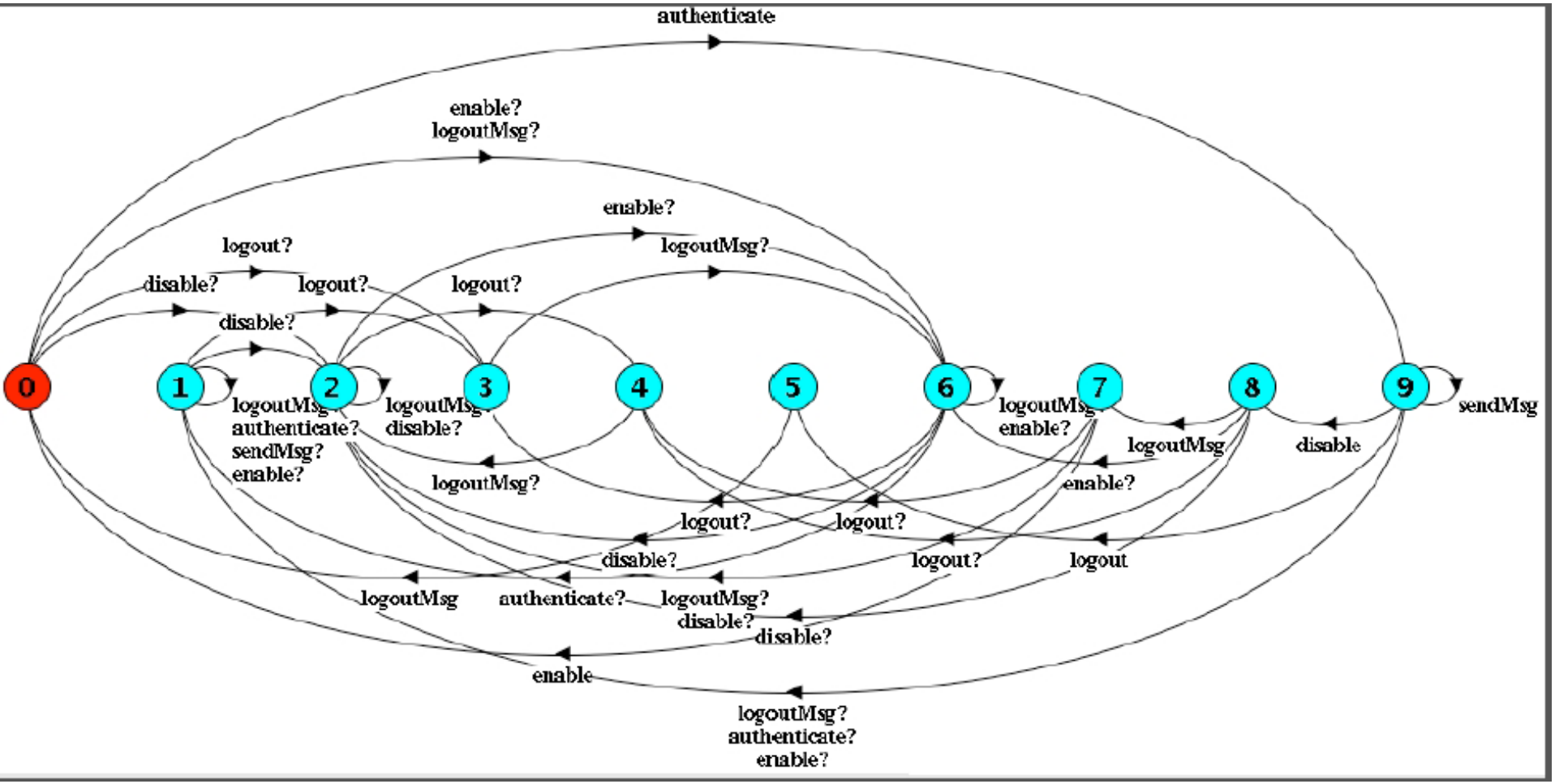
- Synthesis characterises all implementations that satisfy the properties



Synthesis from heterogeneous specifications



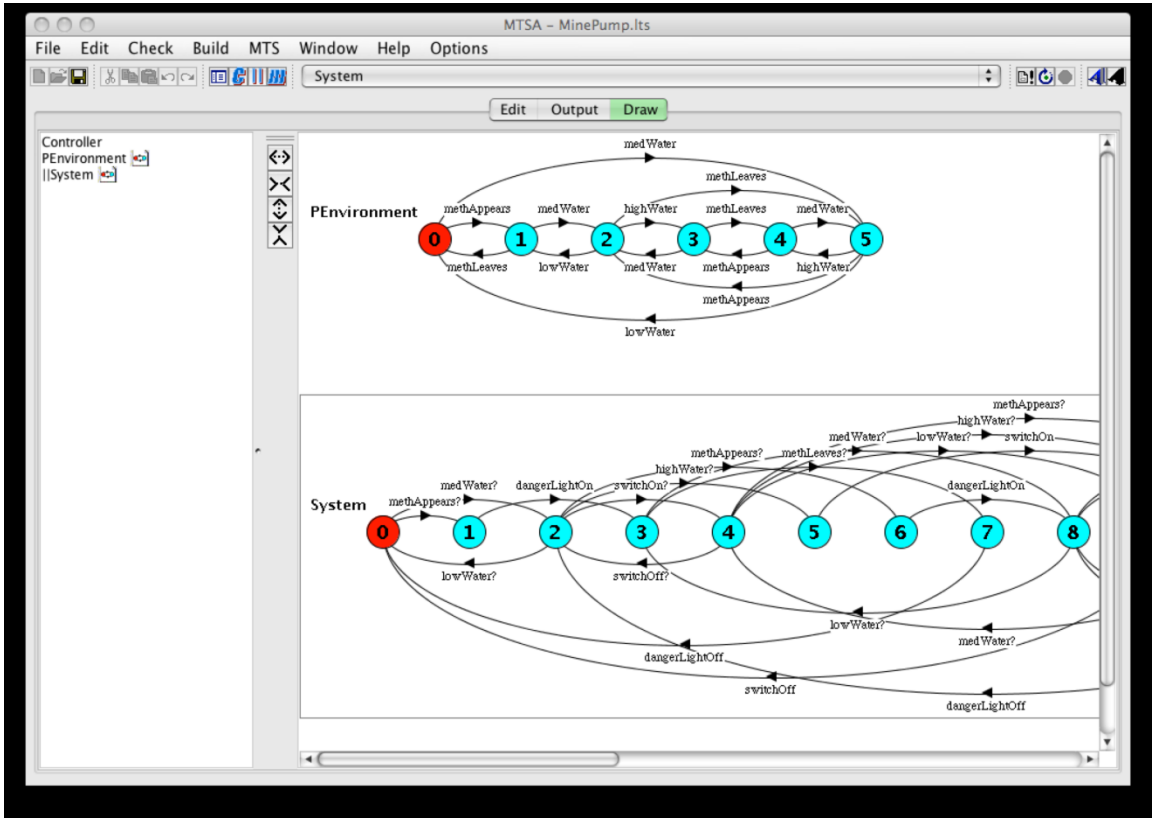
Resulting Model



Tool Support

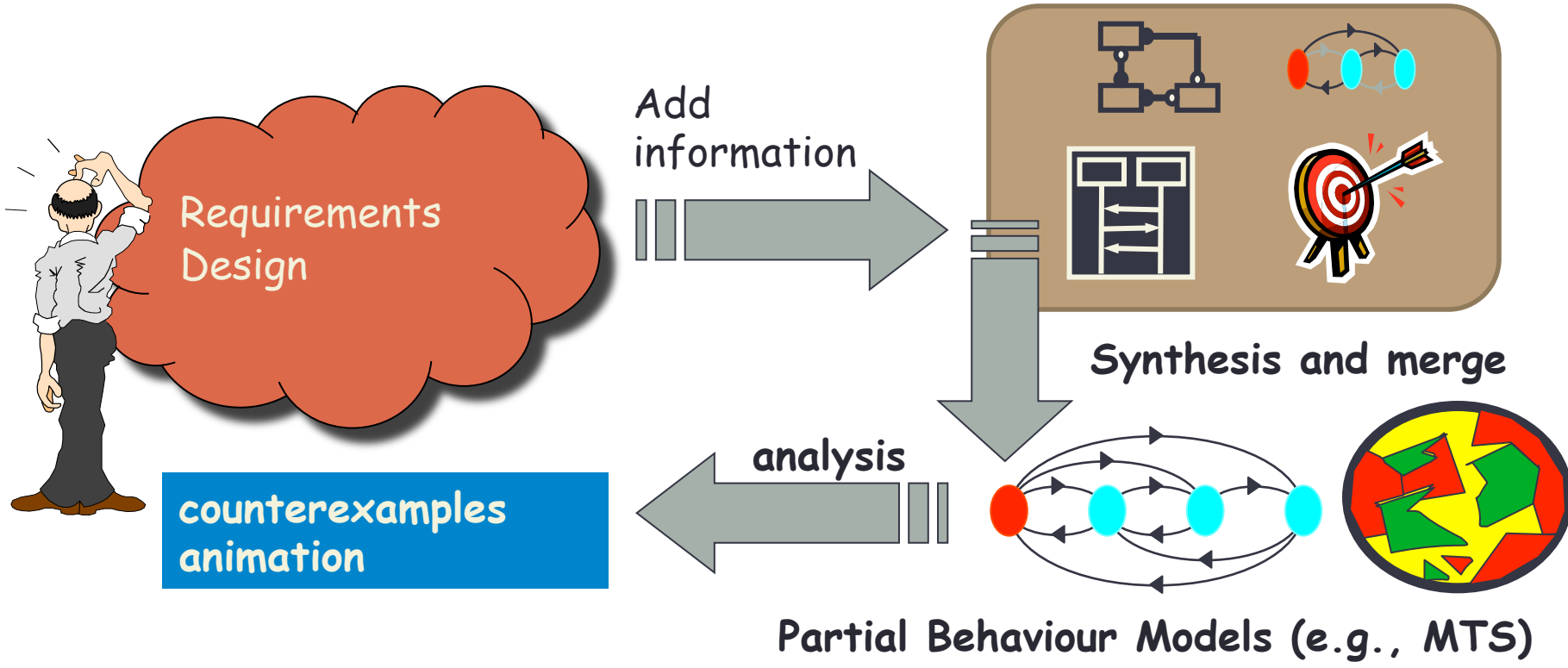
http://sourceforge.net/projects/mtsa/

- Synthesis (Process Algebra, Scenarios, Properties)
- FLTL Model Checking
- Animation
- Composition
  - Parallel
  - Merge
- Others
  - Minimization
  - Hiding
  - Determinization



Summarizing

- Partial behaviour models are the foundations for techniques and tools to develop support for the iterative construction and elaboration of behaviour models

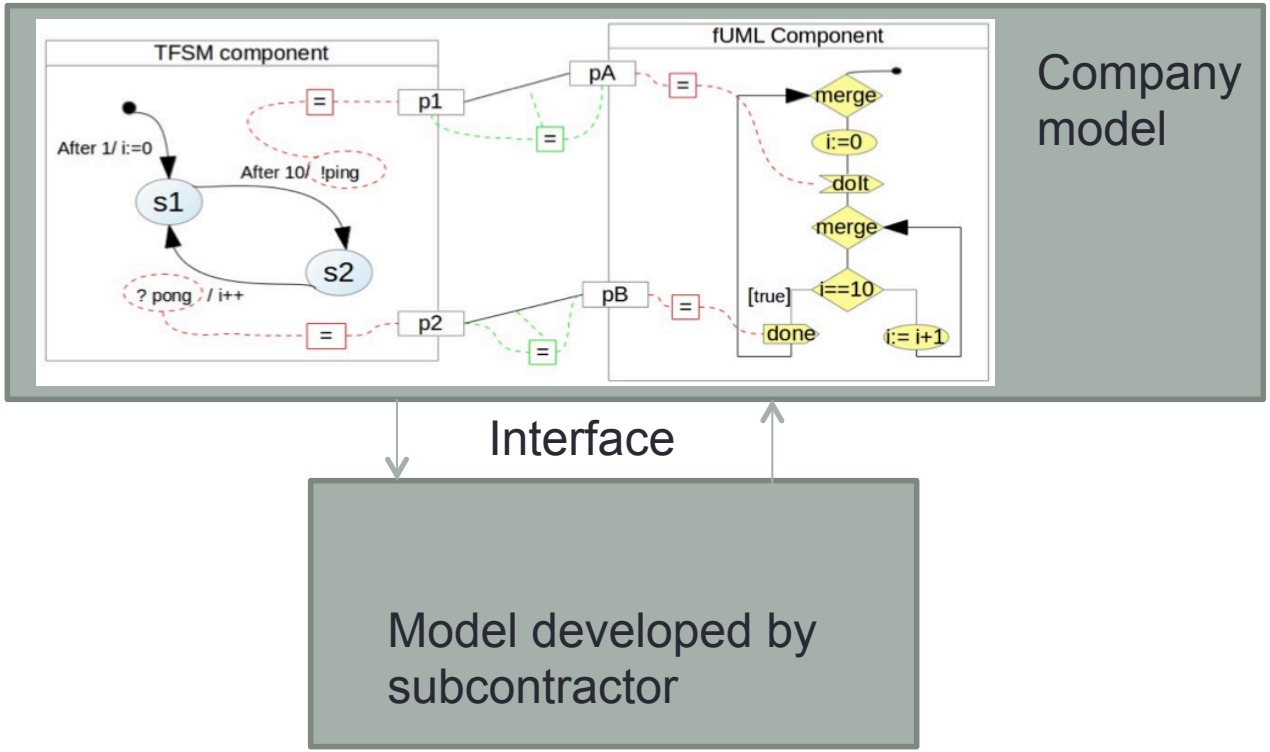


Status of this work

- Toolset for MTS synthesis, verification, animation (mature)
- Recent progress in merging partial models over different vocabularies [CONCUR'13]
  - Requires DMTSs
- Toolset for DMTS analysis (prototype)
- Novel scenario-based specification languages [FSE'11]
- Support for inconsistency resolution [SoSym'11]

Why Relevant to GEMOC?

- We propose to synthesize partial models to construct behavior of missing components, given their contracts, however partial
- The resulting behavioral models can be used, together with existing ones, in testing, animation, verification

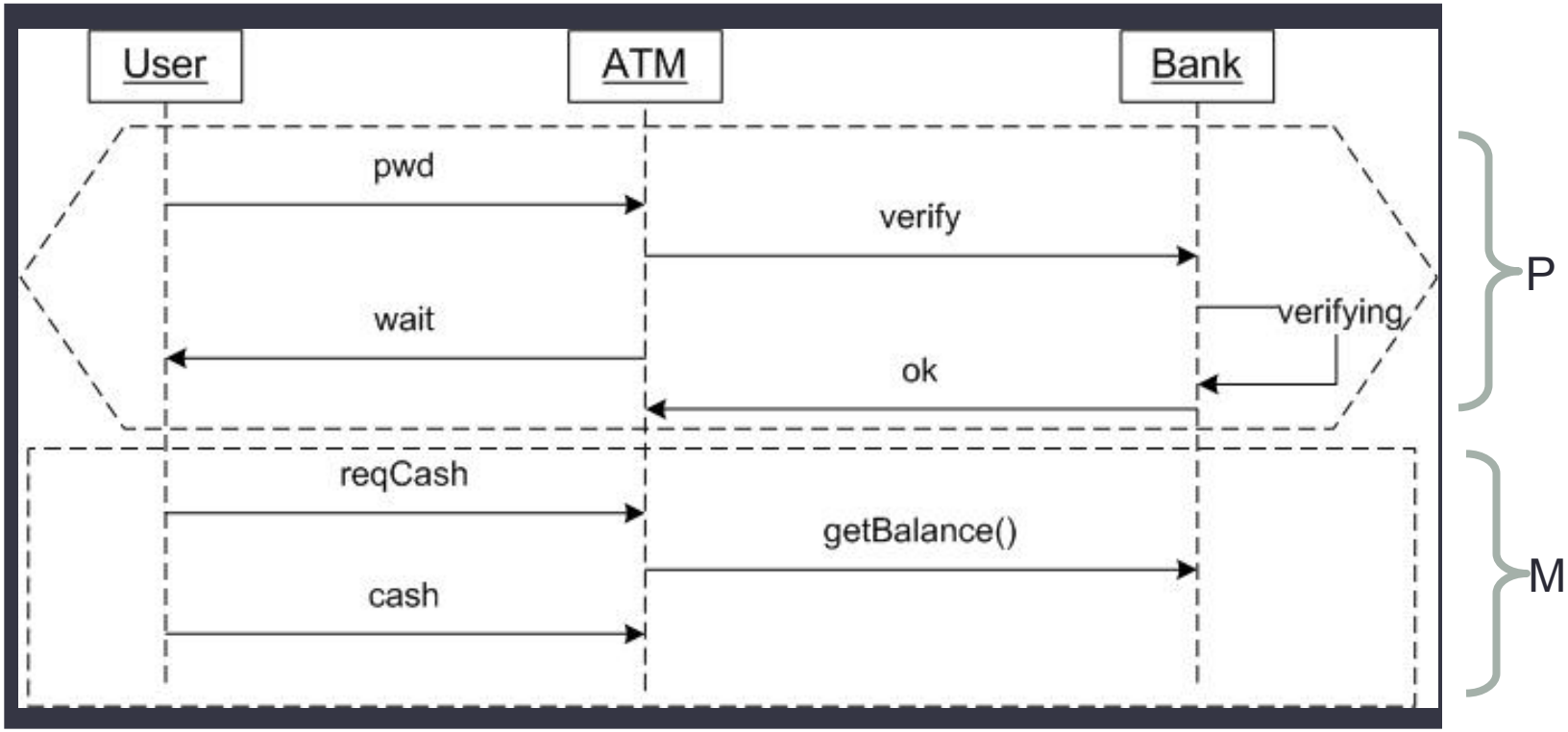


Challenges

- Synthesis to produce a model complying to an assumed meta-model
- Scalability to “large” models
- Timing issues
- Handling relationships between models
- …
- …

QUESTIONS?

Existential Triggered Scenarios



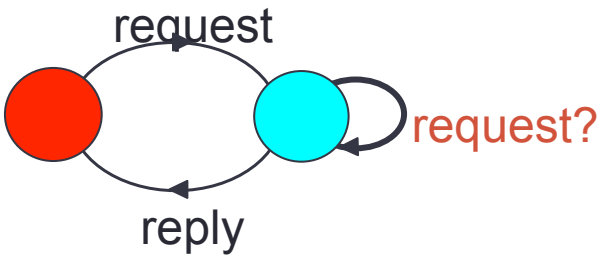
Tree-based Semantics:

When the Prechart holds then, there must exist an execution branch where the Mainchart is next.

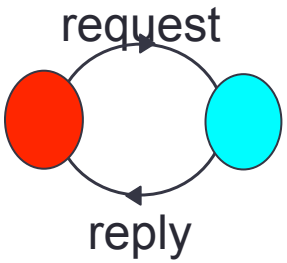
[ICSE'08]

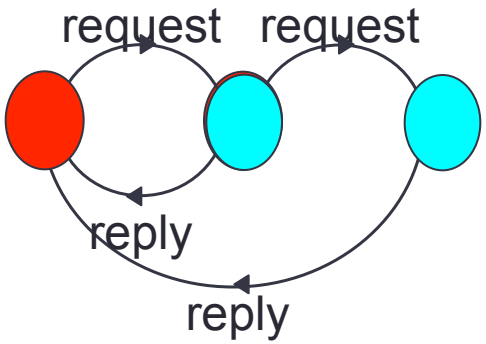
Modal Transition Systems [Larsen88]

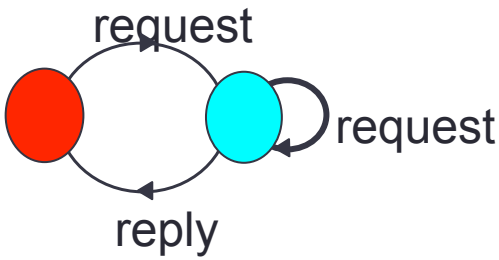
- Extend LTS with an extra set of transitions
  - Required or Must transitions
  - Possible or May transitions



- An LTS L is an implementation of an MTS M if
  - all required behaviour in M is in L, and
  - all behaviour in L is possible in M

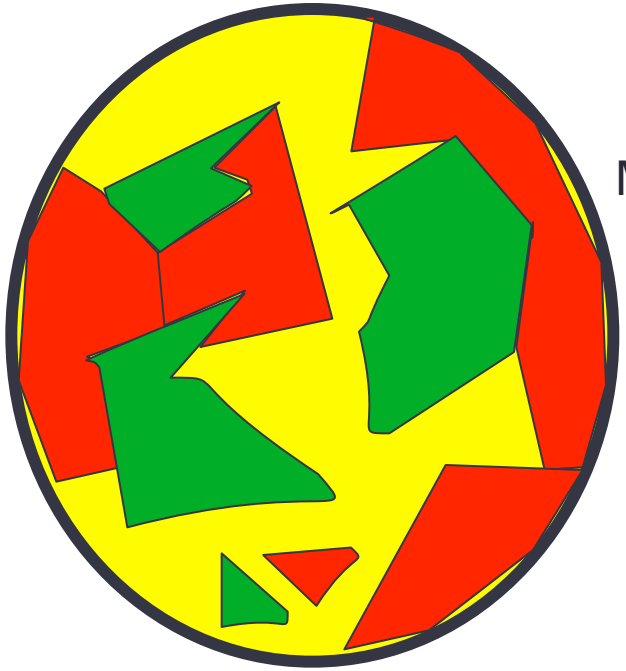






MTS are Sufficiently Expressive

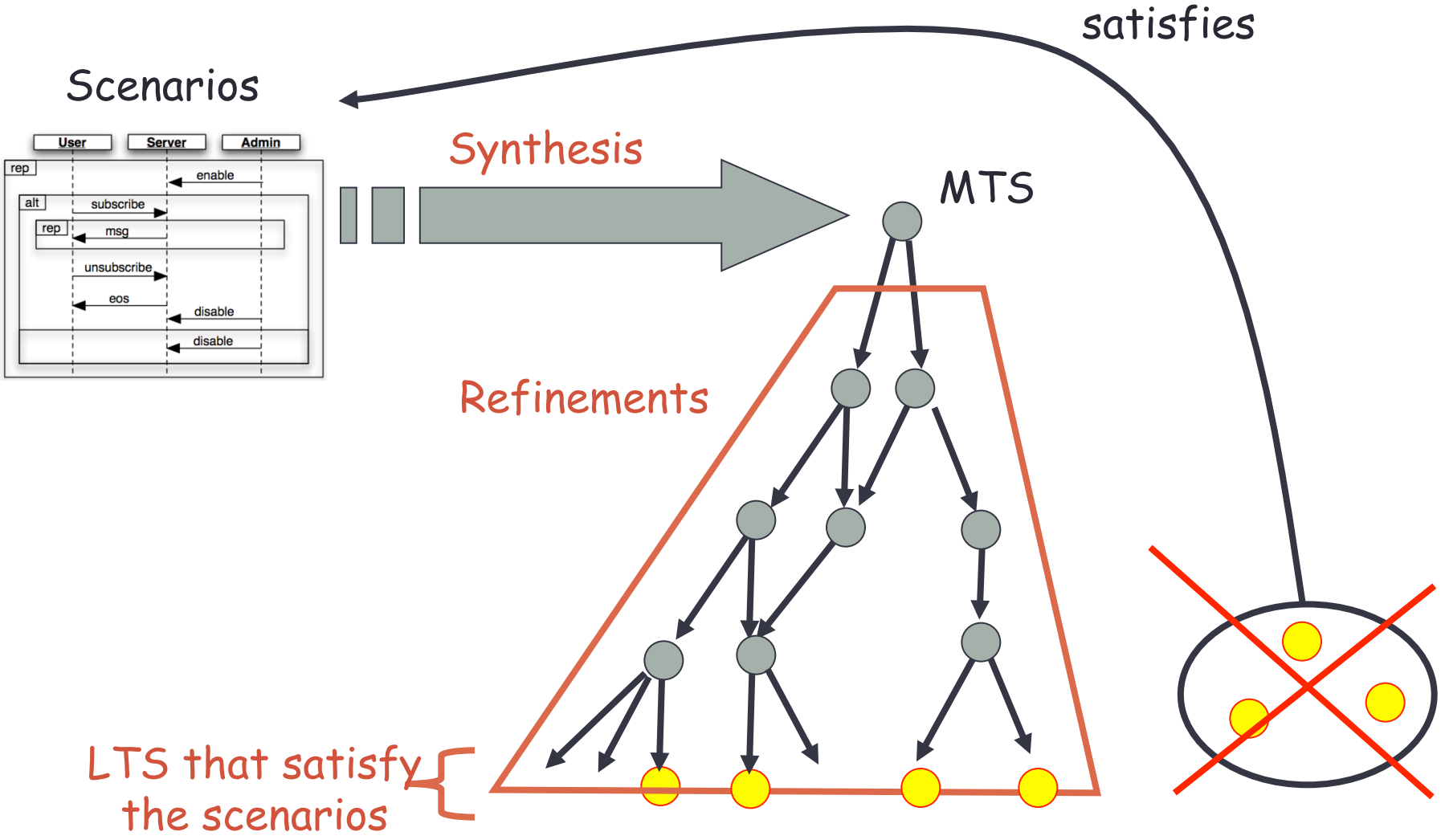
- Modal Transistion Systems can distinguish required behaviour of scenarios, proscribed behaviour from properties and the rest...



MTS from Scenarios and Properties...

Green = MTS required traces
Yellow = MTS possible traces
Red = everything else

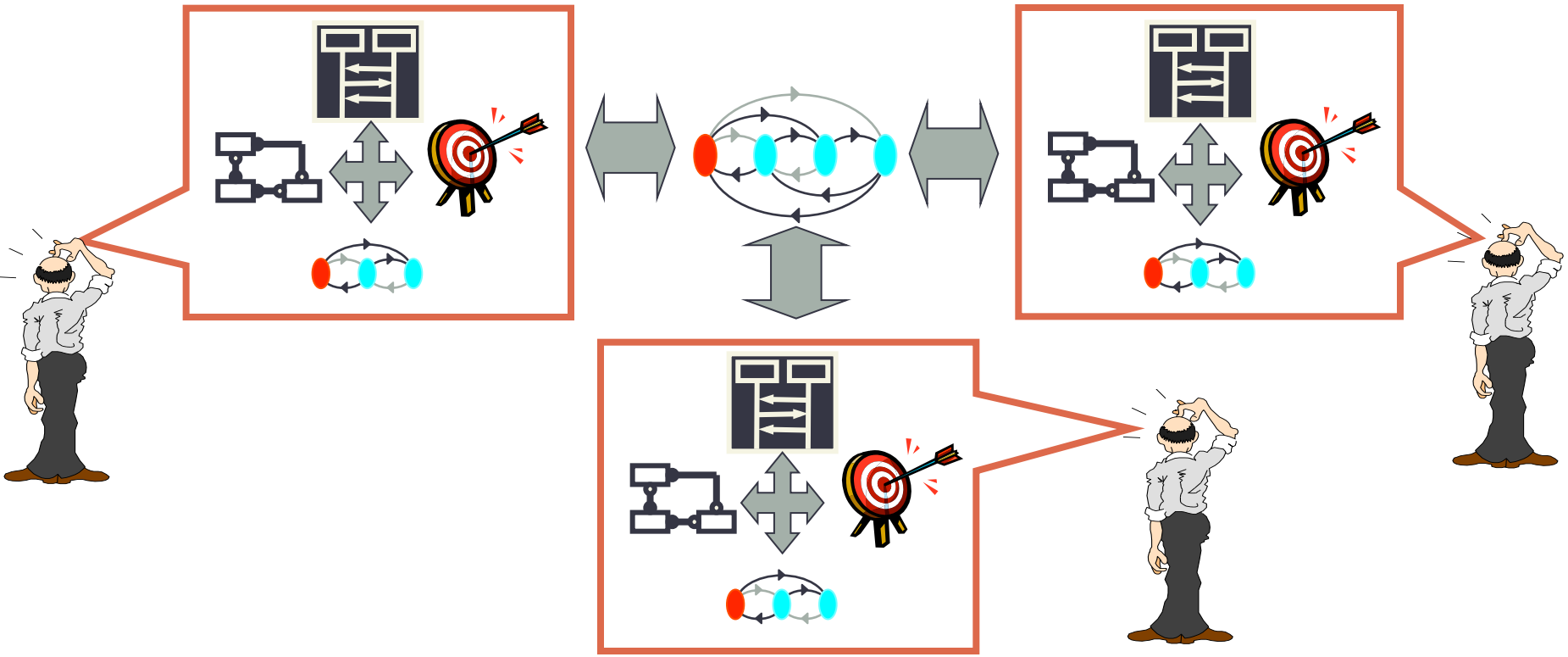
Synthesis from Scenarios



[ICSE07][TSE09]

MTS Merging

- Merging MTS resulting from...
  - Heterogeneous notations
  - Multiple stakeholders



Problems

(D)MTS merge

FSE'11 paper. Language: CSSL

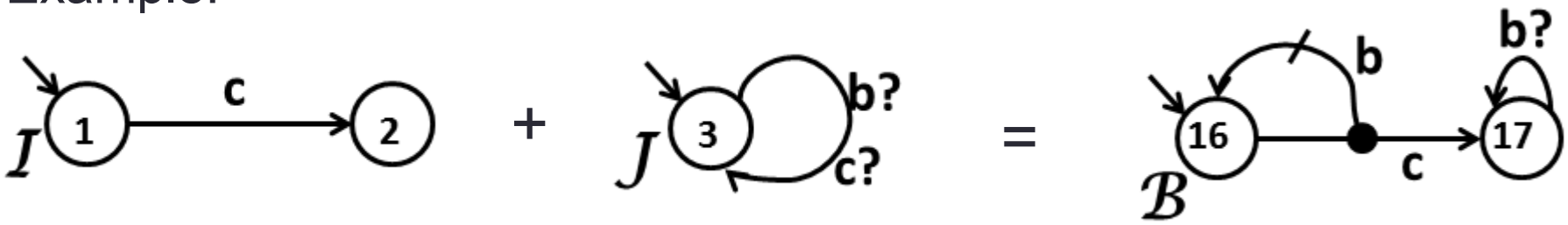
General synthesis of partial models from specs (CSSL?)

Model repair problem

- MTSs over different vocabularies are NOT closed under merge
- What is the 'right' language for capturing properties of behavioural models that come from scenarios?
- This is synthesis w.r.t. safety properties only – what about richer behaviours such as liveness? Timing? Probabilities? Hybrid models? Richer metamodels?
- How to "fix" a behavioral model if it has been determined to be incorrect
  - E.g., some of the sensors fail?
  - E.g., new sensors with new parameters/behavior are detected?
  - E.g., testing determined that that the original properties were "a bit" wrong?

(D)MTS Merge

Example:



- Merge of (D)MTSs is defined when they are ...
  - ... Consistent and
  - ... Compatible
    - each model does not have loops consisting entirely of non-observable operations
    - (Youssef Zaky's masters thesis had a different compatibility condition)
- CONCUR'13 paper forthcoming (with Shoham Ben-David and Sebastian Uchitel)
  - See poster
- Development of DMTSA (on top of MTSA) in progress