Temporal Action Language (TAL): A Controlled Language for Consistency Checking of Natural Language Temporal Requirements

(Preliminary results)

University of Kentucky

Department of Computer Science

Wenbin Li, Jane Huffman Hayes, Miroslaw Truszczynski

NFM 2012, April 3, 2012

This work is funded in part by the National Science Foundation under NSF grant CCF-0811140 and by NASA JPL under grant 1401954.

UK

The Problem

- Faulty requirements cause
  - 60% – 80% of project failures [1]
  - 80% of all software rework [2]

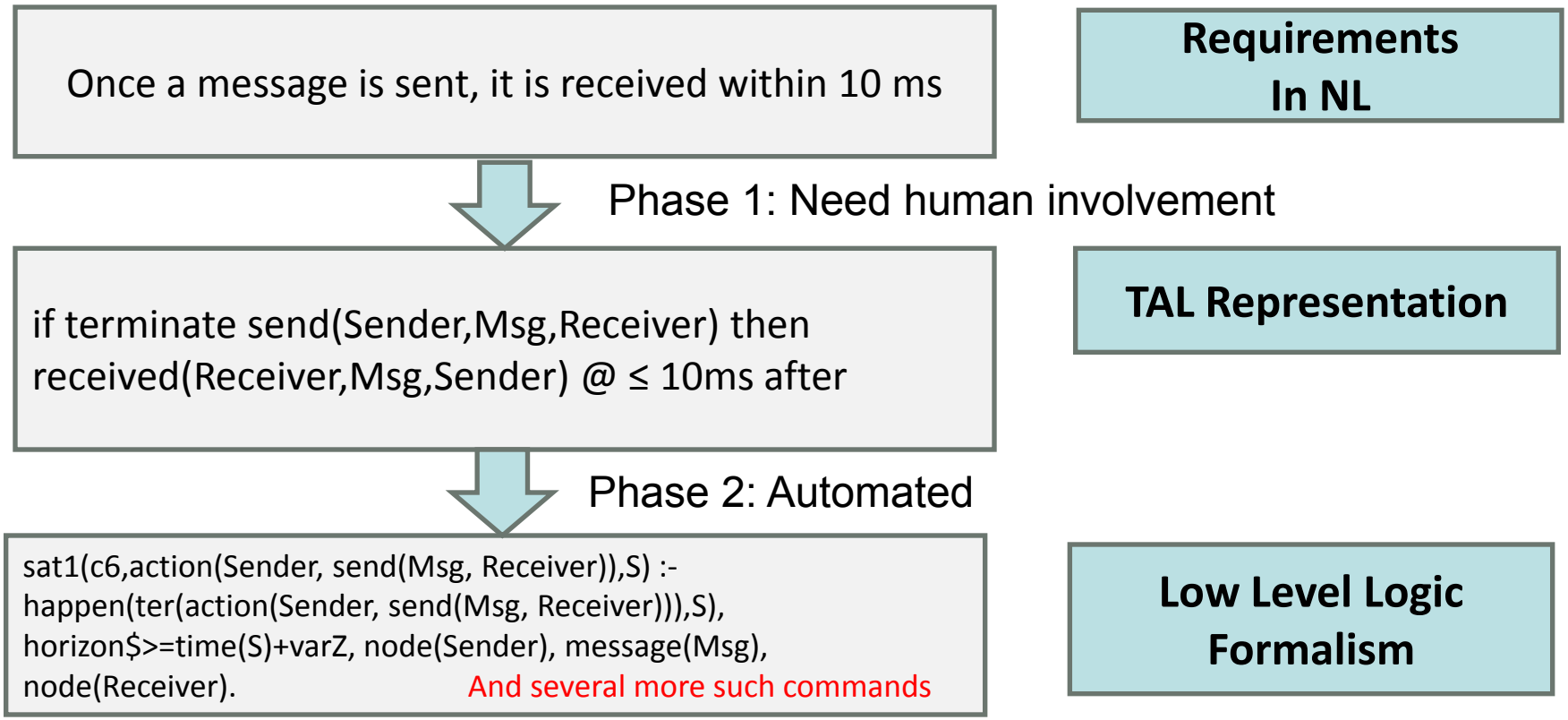
- Inconsistency : a common requirement fault
  - System contains nodes and a server.
  - Nodes A, B, and C must connect to the server *within the first 5 seconds*.
  - Connections cannot be established *simultaneously*.
  - It takes *2 seconds* to establish a connection.

Our Goal

- Detect inconsistencies
  - Focus on requirements stated in *natural language*
  - Focus on *temporal constraints*
- Minimize human involvement
- A language formal and human readable

Proposed Approach

- Temporal Action Language **TAL** as bridge



Temporal Action Language (TAL) Syntax

- Syntax
  - Declarations
    - sort node
    - fluent connected(nodeA, serA)
    - action sendMsg(nodeA)
  - Action Definitions: reuse AL (Baral, Gelfond 2000)
    - connect(*serA, nodeA*)

      causes *connected(nodeA, serA)* **if** *systemOn*
    - **impossible** *write(nodeA, serA)*

      if not *connected(serA, nodeA)*

Temporal Constraints

- < prompt | fluent > @ < when >
  - prompt: **commence** *Act* **terminate** *Act*

- Examples
  - **terminate** *dropConn(serA, nodeA)*
    @ ≤ 15 second **after startTime**

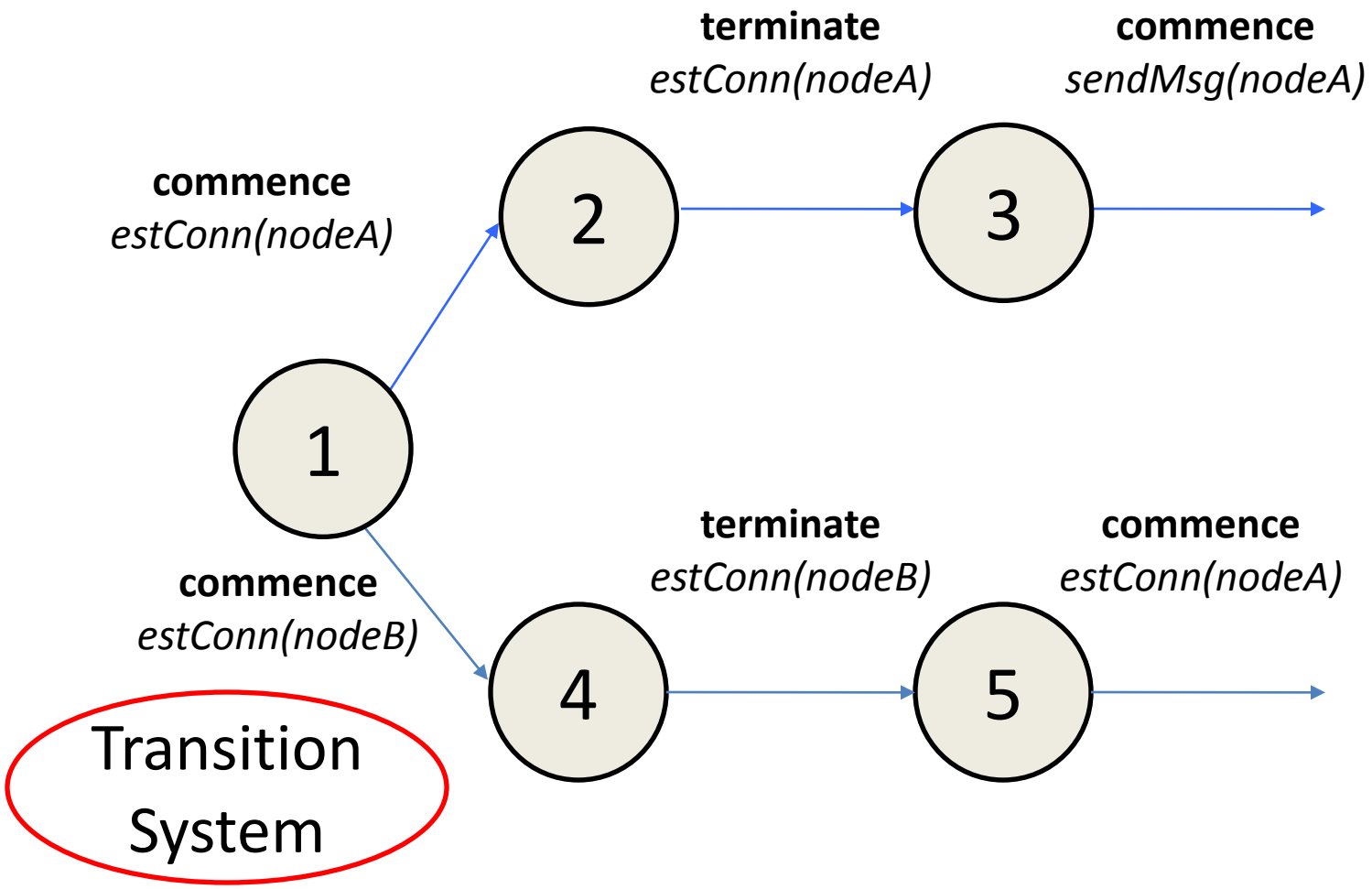
  - **commence** *sendMsg(nodeA)*
    @ ≤ 5 seconds **after terminate** *estConn(nodeA)*

  - *received(serA,message,nodeA)*
    @ ≤ 10 millisecond **after terminate** *send(nodeA, msg, serA)*

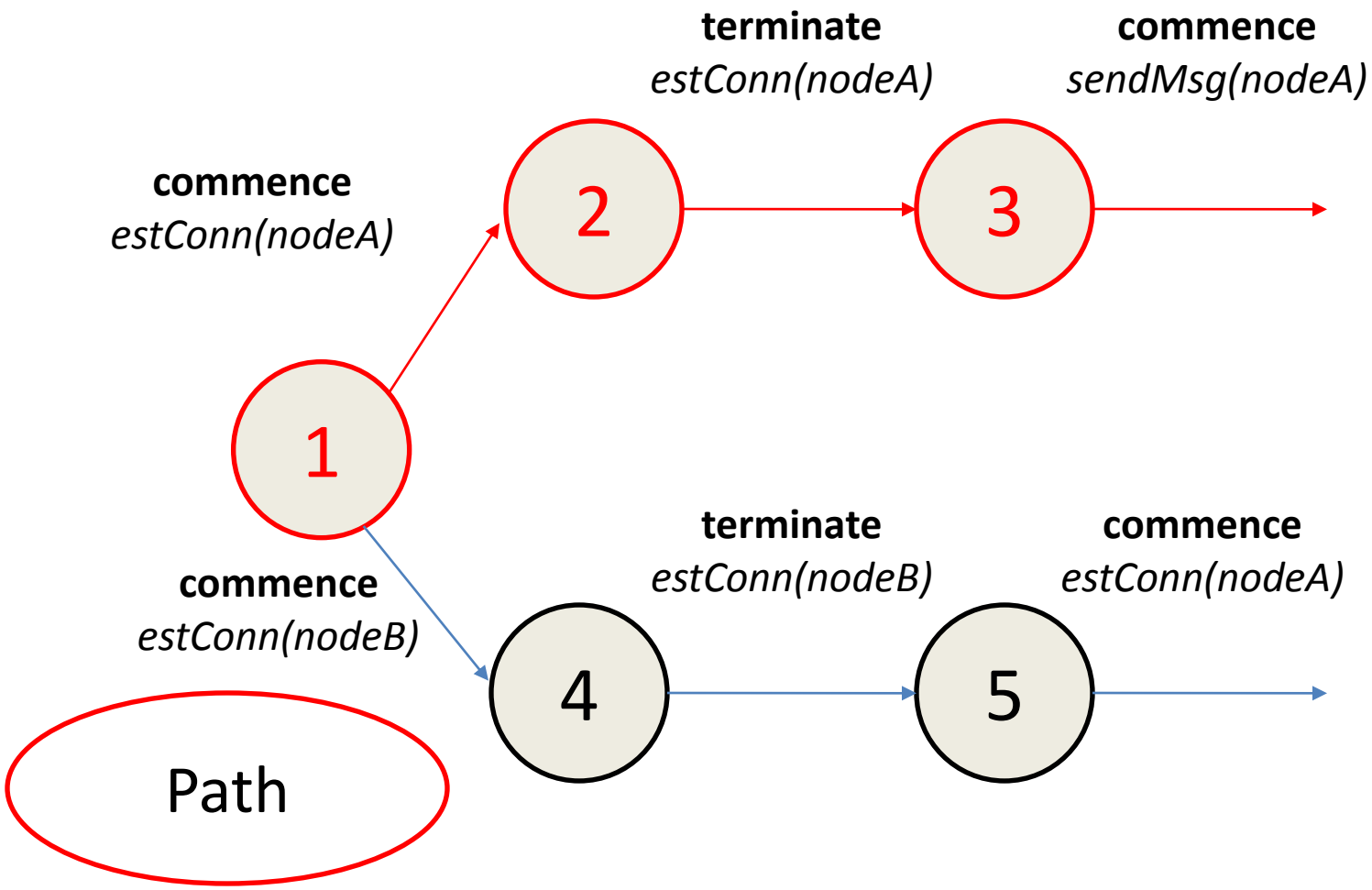
Temporal Action Language (TAL) Semantics

- Semantics
  - Transition system
    - States determined by fluents
    - Arcs labeled with prompts or “Time"
    - Illustrate all possible ways for system to evolve
  - Path
    - A particular scenario
  - Timed path
    - A scenario labeled with times when changes occur

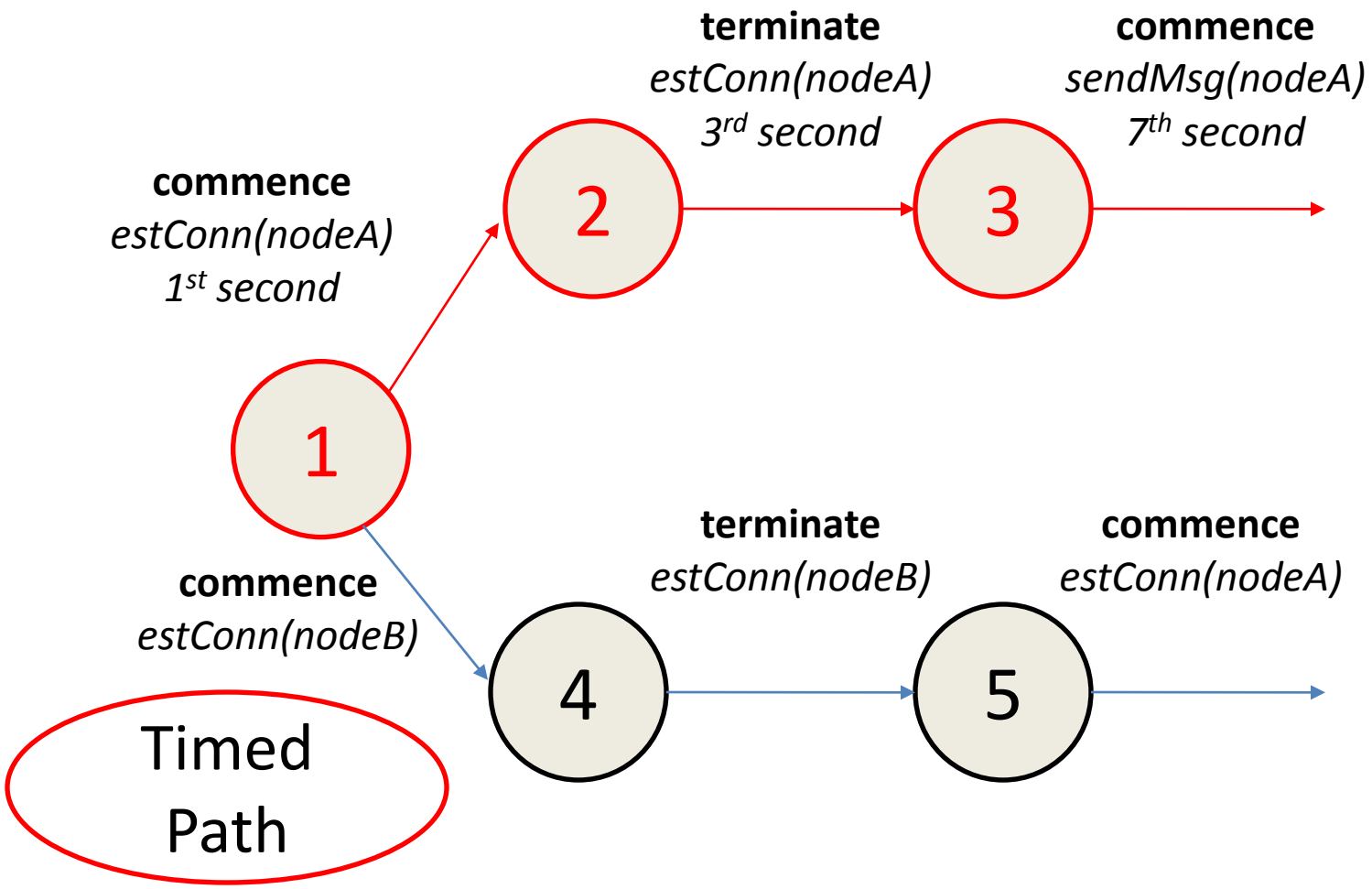
Temporal Action Language (TAL) Semantics



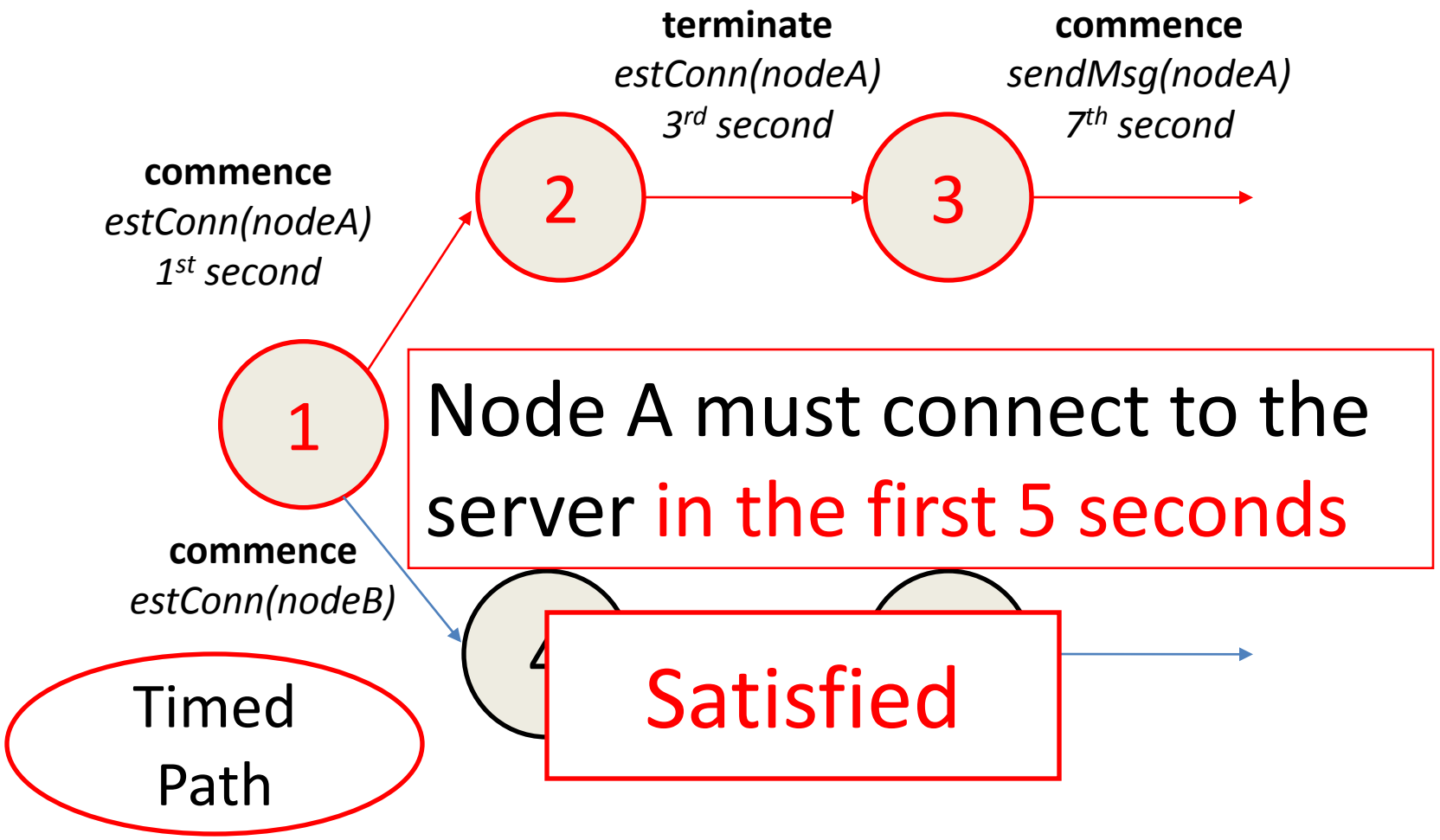
Temporal Action Language (TAL) Semantics



Temporal Action Language (TAL) Semantics



Temporal Action Language (TAL) Semantics



Temporal Action Language (TAL) Consistency Checking

- Valid timed path P
  - All temporal constraints satisfied along P
- Consistency
  - Arbitrarily long valid timed paths exist
  - Arbitrarily long system runs possible
- Limited consistency
  - Timed paths up to a certain time horizon
  - No inconsistency shows up prior to the horizon

Temporal Action Language (TAL) Consistency Checking

- To detect inconsistency search for a valid timed path
  - Translate TAL into low-level formal system
  - Use reasoning tools of that system

Validation

- So far:
  - Use NASA CM1 requirements
  - Generate a TAL theory for a subset of CM1 set that includes all temporal requirements
  - Translate that TAL theory into answer set programming (ASP)
  - Use ASP solver clingcon for processing

Related Work

- Software Cost Reduction
  - Heitmeyer, Labaw, and Kiskis, 1995 [6]
- Timed automata and temporal logic
  - Knapp, Merz, and Rauh, 2002 [7]
  - Bengtsson, Larsen, Larsson, Pettersson, and Yi, 1996 [8]
  - Bozga, Daws, Maler, Olivero, Tripakis, and Yovine 1998 [9]
- UML
  - Selic, 1998 [10]

Future Work

- From natural language to TAL
  - Partially automate generation of TAL theories from natural language text
- From TAL to low level logic formalism
  - ASP (Completed, but more work on translation optimization and performance needed)
  - Temporal logics
  - Timed automata
- Systematic experiments
- Seek broad feedback on TAL readability, design

References

[1] Bob Lawhorn, March 2010

[2] Critical Logic newsletter, 2011

[3] C. Baral and M. Gelfond, “Reasoning agents in dynamic domains,” Logic-based artificial intelligence, pp. 257–279, 2000

[4] V.W. Marek and M. Truszczyński, “Stable models and an alternative logic programming paradigm,” *The Logic Programming Paradigm: a 25-Year Perspective*, 1999, pp. 375-398.

[5] M. Gebser, M. Ostrowski, and T. Schaub, “Constraint answer set solving,” International Conference on Logic Programming (ICLP), pp. 235–249, 2009.

[7] C. Heitmeyer, B. Labaw, and D. Kiskis, “Consistency checking of scrstyle requirements specifications,” in Proceedings of the 2nd IEEE International Symposium on Requirements Engineering. IEEE Computer Society, 1995.

[8] A. Knapp, S. Merz, and C. Rauh, “Model checking-timed uml state machines and collaborations,” Lecture notes in computer science, p. 395C416, 2002.

[9] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, “Uppaal: a tool suite for automatic verification of real-time systems,” LECTURE NOTES IN COMPUTER SCIENCE, pp. 232–243, 1996.

[10] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine, “Kronos: A model-checking tool for real-time systems,” LECTURE NOTES IN COMPUTER SCIENCE, pp. 298–302, 1998.

[11] B. Selic, “Using uml for modeling complex real-time systems,” Logicbased artificial intelligence, pp. 250–260, 1998.

Thank you!

Questions?