Finite State Transducers for Policy Evaluation and Conflict Resolution

Javier Baliosian and Joan Serrat

Universitat Politècnica de Catalunya – Spain
Network Management Group

jbaliosian@tsc.upc.es

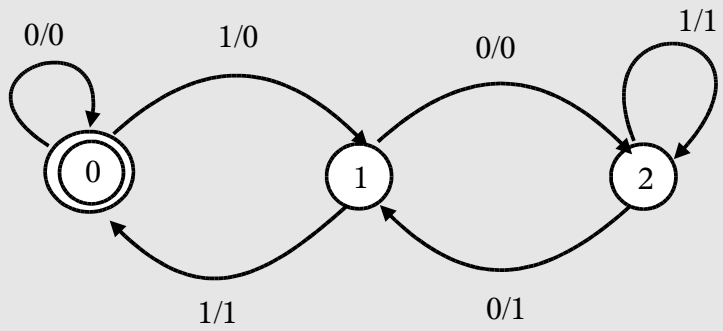
Motivation

A policy conflict occurs when the conditions of two or more policy rules that apply to the same set of managed objects are simultaneously satisfied, but the actions of two or more of these policy rules conflict with each other. (Strassner)

- We have been looking for a technology independent model for conflict resolution in PBNM.
- In fields such as speech recognition, quick decisions based on ambiguous grammatical rules are required. They make use of Finite State Transducers (FSTs).
- We propose an approach consisting of an adapted subset of FST concepts with the aim of detecting and solving conflicting policy rules.

Finite State Transducers

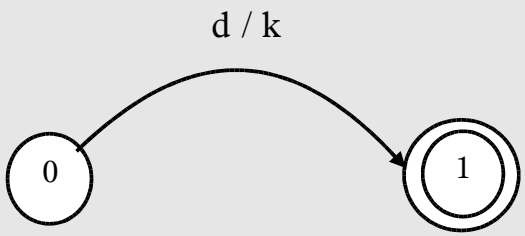
- They are special automata for which, on each edge, there are two labels instead of one.
- Transducers can be seen as devices defining a class of relations over strings of symbols.
- Their implementations tend to have good performance.



Classic transducer representing division by 3

Obligations

- They are modeled as a graph with one edge only
- The incoming label is the condition part
- The outgoing label is the action
- It must be added to the existing model with a *union*



if the user dials up
then execute the action
connect

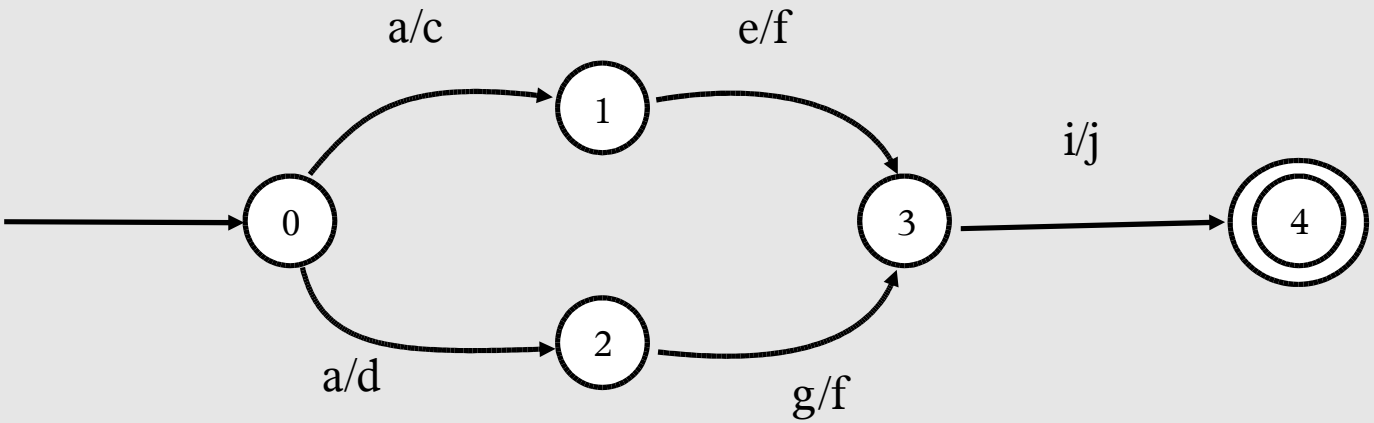
FST Operations

Operations on FSTs

- Union
- Intersection
- Complement
- Composition
- Kleene closure
- Determinization

- In the general case, FSTs are **not** closed under some of these operations, but
- The restrictions required for the operations' closure are compatible with PBNM.

Determinization

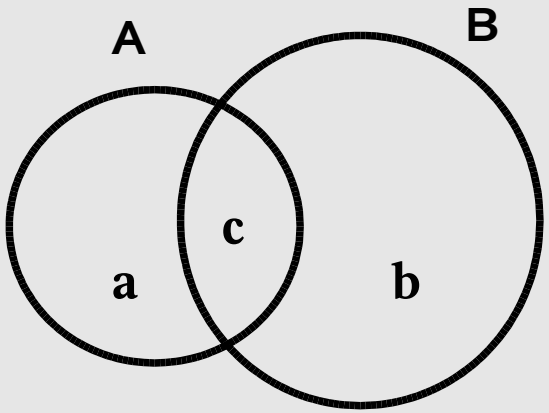


Ambiguous transducer

- Determinzation is the actual conflict resolution process
- It consists of leaving only one possible edge to choose in a given node
- For this algorithm, we replace the classic labels for Tautness Functions.
- We named the extended transducers as TFFSTs

Tautness Functions

- This is intended to represent how "taut" a condition is around an event
- Related to the concepts of "distance" and "nested domains"
- It assigns a *real in* [-1,1] to the duple <condition,event>



$$\tau_A(a) \geq 0$$
$$\tau_A(b) < 0$$
$$\tau_A(c) < \tau_B(c)$$
$$\tau_{A \vee B} = max(\tau_A, \tau_B)$$
$$\tau_{A \wedge B} = min(\tau_A, \tau_B)$$
$$\tau_{\neg A} = -\tau_A$$

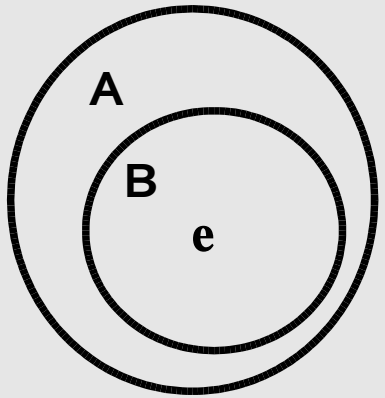
$$\tau_{A \rightarrow_\tau B} = \begin{cases} \tau_A, & if\ \tau_A < \tau_B \\ -1, & else \end{cases}$$

$$\tau_{A \rightleftarrows_\tau B} = \begin{cases} \tau_A, & if\ \tau_A = \tau_B \\ -1, & else \end{cases}$$

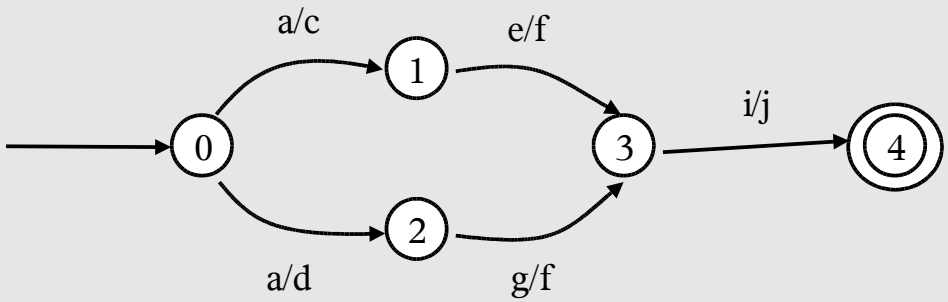
Examples of TFs

- The most straightforward example is when a domain is "inside" another domain. *B* is tauter than *A* on the event *e*.
- *In PCIM we can count the number of elements in PolicySetAppliesToElement association. The more elements there are, the less taut the condition.*

Policy A: *All users are forbidden to reboot workstations.*
Policy B: *The system administrators are authorized to reboot workstations.*
Event e: A system administrator is trying to reboot a workstation

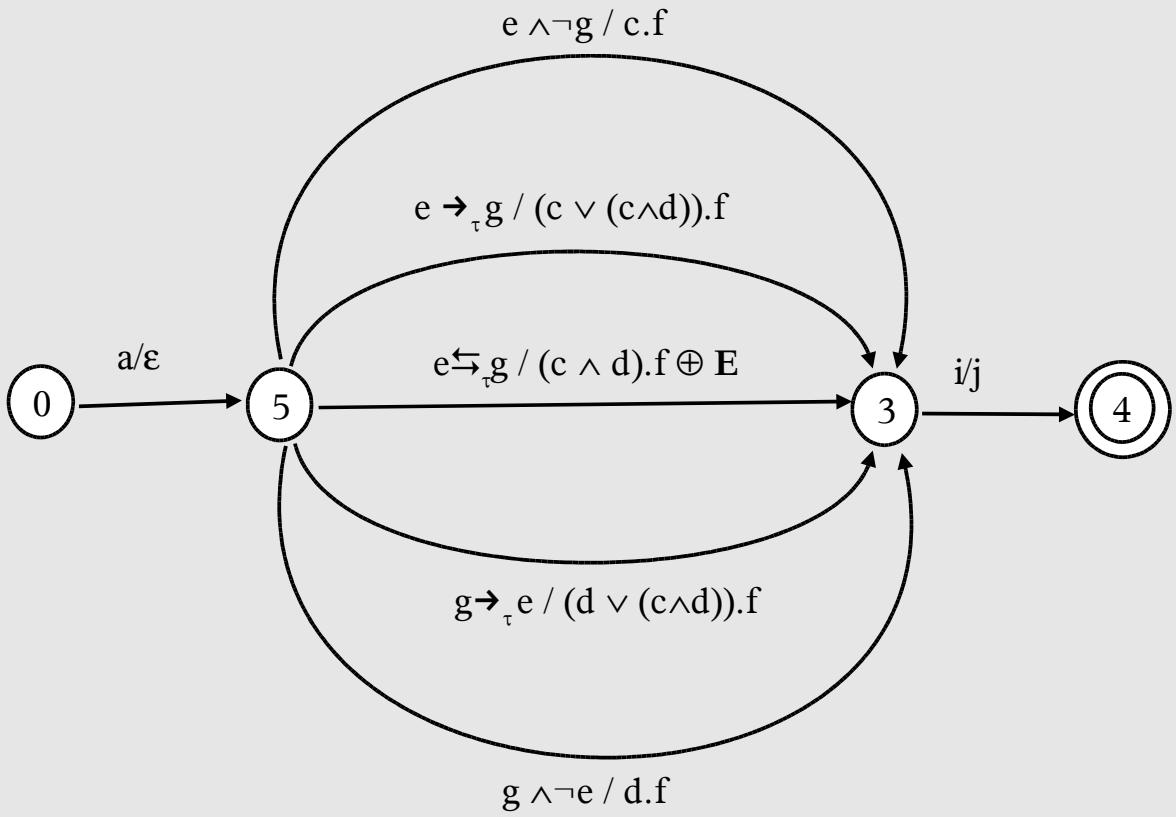


Determinization (cont'd)



Before

After



TFFST Semantics in Policy-Based Management

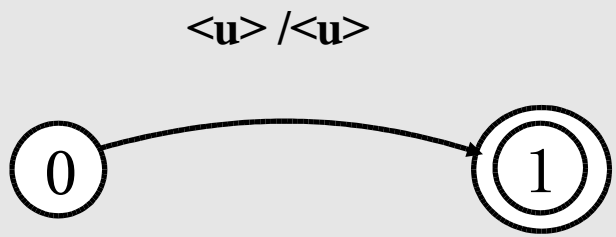
- Rights
- Prohibitions
- Obligations
- Dispensations
- Constraints
- Conflict Resolution

Rights

- These are just an edge with the same label on both sides and the identity flag on.
- This means: Every time input is positive under “u,” then it replicates the input on the output.

Example:

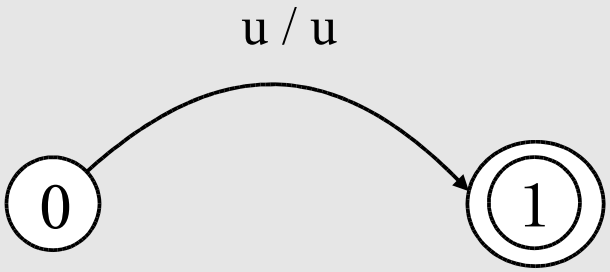
Rule 1: *The users are authorized to print*



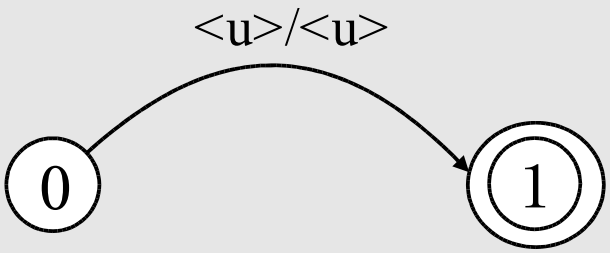
T1

Identities on TFFSTs

- Both labels on each edge of a transducer can be seen as a condition that may be fulfilled by several events.
- Identity is needed to reproduce exactly the input on the output.



Any action that fulfills "u" may be produced



The same incoming event is thrown as output if it fulfills condition "u"

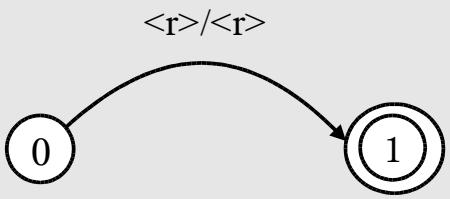
Prohibitions

- These are expressed as the substraction of a right

Example:

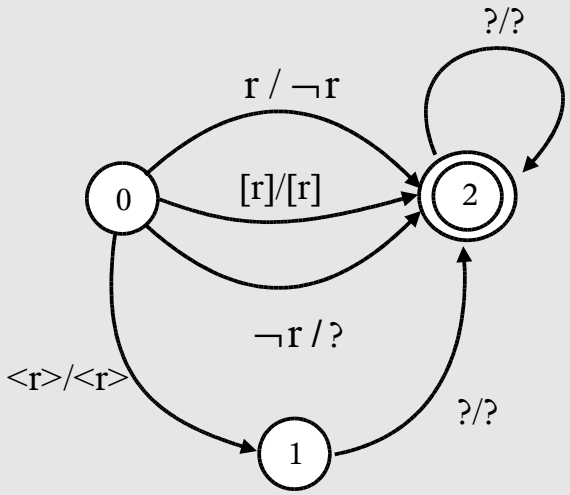
Rule 1: *The users are authorized to print (T1)*
Rule2: Guest users are forbidden to print

T2



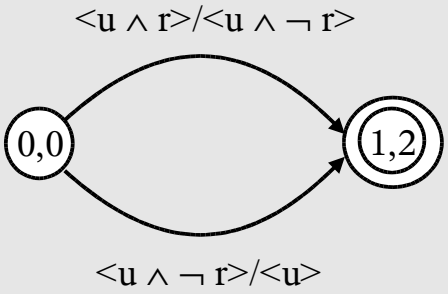
Auxiliary right:
Guest users are ***allowed*** *to print*

$\overline{T2}$



T2 Complement

$T1 \cap \overline{T2}$



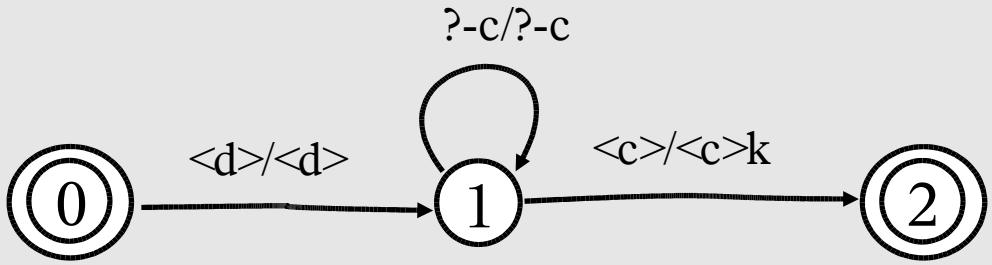
$T1 - T2$

Obligations

- Obligations can be associated with more than one event

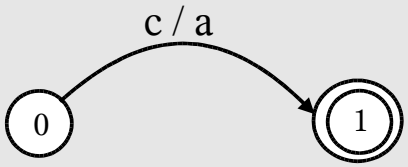
Example:

Rule 1: *if the user dials up (d in the figure) and the system sends the order of charge (c in the figure), then the connect action (k in the figure) should be executed. (Chomicki et al. example)*

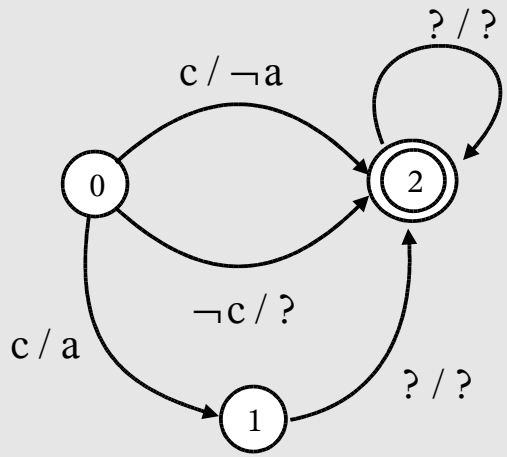


Dispensations

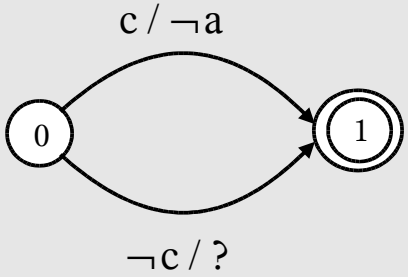
- As in the case of prohibitions, dispensations should be expressed as the intersection of the existing policy body with a transducer expressing the complement of an obligation.



Auxiliary obligation



Obligation's complement

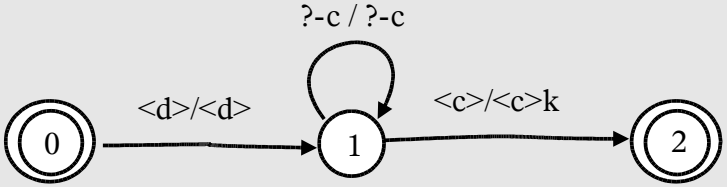


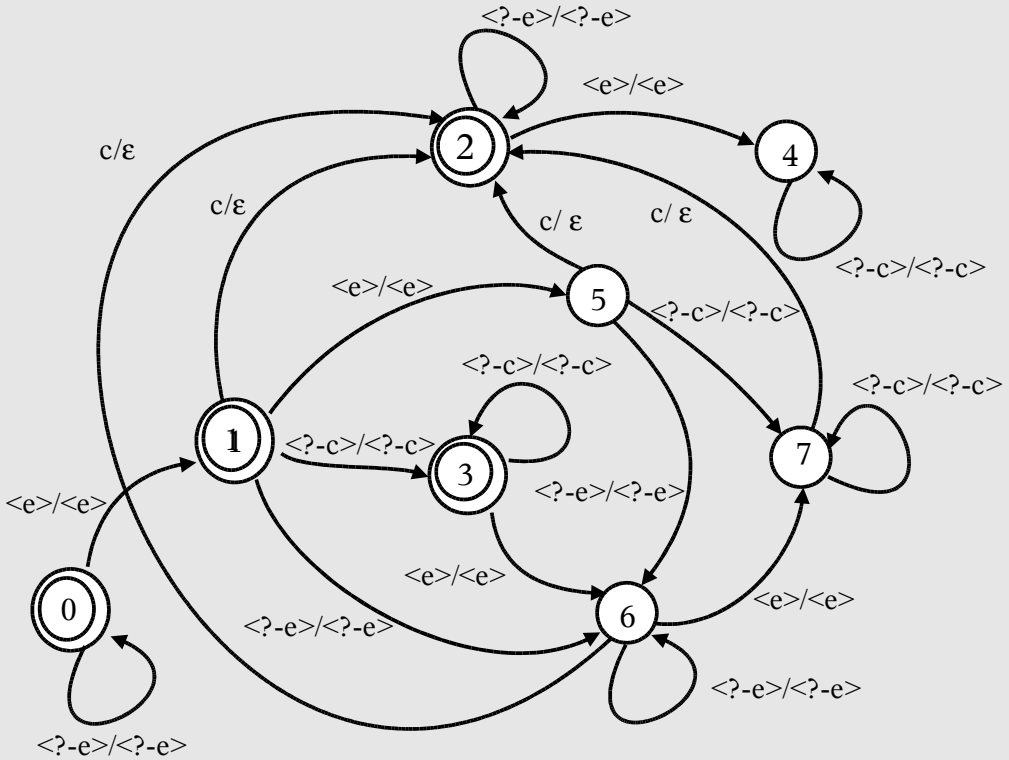
Dispensation in an “all permitted” environment

Constraints

- Constraints must be *composed* after the model of policies.

Example:



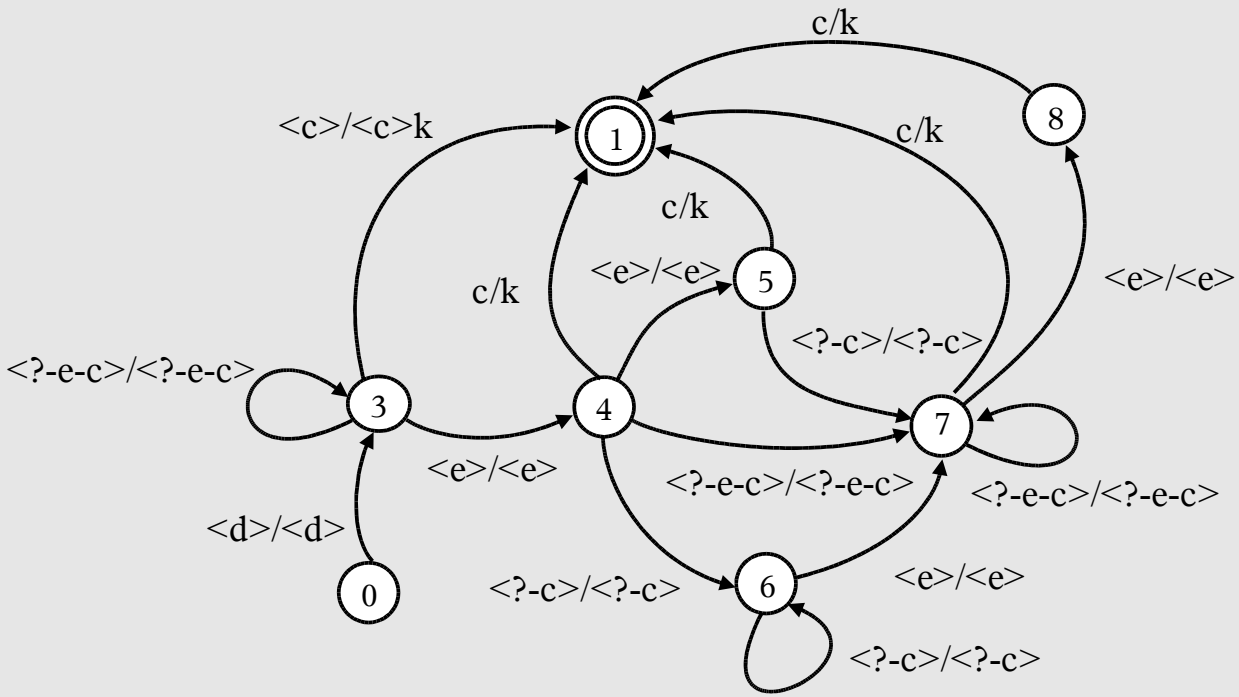


Rule 1: *if the user dials up and the system sends the order of charge, then a connect action should be executed.*

Rule 2: *if an error (e in the figure) occurs, charge action should not be triggered*

Constraints (cont'd)

Example continuation:
Composition of both transducers is



- This TFFST is equivalent to the sequential evaluation of the last two.

The Overall Process

1. Compute the union of all transducers representing rights and obligations
2. Subtract the transducers representing prohibitions and dispensations
3. Compose the resulting transducer with each constraint transducer
4. Determinize the resulting transducer to solve conflicts

Conclusions

- A **formal model** based on a new entity called TFFST was developed for conflict detection and resolution of **modality conflicts** and some **dynamic conflicts** by means of constraints.
- Its operations do not solve problems that were not solvable before but,
- our framework is designed to be **efficient** and **independent of technology**.
- Conflict resolution is carried out beforehand, and **runtime processes have a linear order** on the amount of incoming events.
- The model takes advantage of experience from other fields.

Conclusions (cont'd)

- Tautness Functions are an **abstraction layer**
  - They make the conflict resolution process as general as possible.
  - From the point of view of the algorithms, they are **technology-independent**.
  - They make it possible to deal with **orthogonal conditions**.
  - They are a research issue in themselves.

Ongoing and Future Work

- Modeling with ***weighted TFFSTs*** for explicit priorities.
- Support for more than one PDP.
- Scalability evaluation.
- Development of practical network management related Tautness Functions.
- Rewriting of ***determinization*** algorithm.
- Policy re-writing from TFFSTs must be researched.