A Transformation of SDL Specifications – a Step towards the Verification

Natalia Ioustinova[1] and Natalia Sidorova[2]

[1] Inst. für Technische Informatik
Universität Rostock
Germany
ustin@informatik.uni-rostock.de

[2] Dept. of Math. and Computer Science,
Eindhoven University of Technology,
The Netherlands
n.sidorova@tue.nl

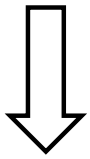
Outline

- Motivation
- SDL time semantics
- Timer transformation
- Timer abstraction
- Conclusions

Model Checking in Practice

Model checking problem:
Does system S satisfy property f ?

Method: *state-space exploration*



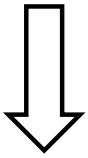
Pros: *a "push-button technology"*
and
Cons: *applicability to relatively small finite state systems only*

---

abstractions,
compositional techniques,
partial-order reduction…

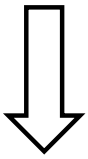
A Trade-off:

A limitation of model-checking:
its applicability to relatively small finite-state systems only



Input languages of model-checkers are rather "primitive"

The specification language should provide a framework for constructing "natural" descriptions of systems



Specifications written in "designer-friendly" languages are hard to verify

SDL

(Specification and Description Language)

Specification of telecommunication software, aircraft and train control, medical and packaging systems.

Theoretical Model:
a set of communicating extended state machines (processes) that run in parallel.

The input port of the process:
an unbounded FIFO queue.

Communication is based on the signal exchange between the system and its environment and between processes within the system.

Scheduling is critical !

Concept of Timers in SDL

Time and **Duration**: predefined data types

NOW operator gives access to the global system time

Timer activation: **SET(NOW** + delay, T)

Timer deactivation: **RESET**(T)

Timeout transition: enabled if T equals **NOW**.

It sends a timeout signal to the process queue.

Timeout signals are processed like normal signals.

Set or Reset of a timer whose timeout signal is in the input queue ⇒

remove the timeout signal from the queue.

Time Semantics

- Discrete time
- Transitions are instantaneous
- Time can only progress if the system is blocked, i.e., all the input queues are empty, and no NONE input is enabled. The only possible actions are time progress and input from the environment.

$$\text{NOW} \rightarrow \text{NOW} + 1$$
until some active timer expires

- In case if several time-out transitions become enabled at the same time, one of them is taken (non-deterministically) to unblock the system and the rest are taken later at arbitrary points of the time slice.

What It Means for the Verification

- NOW is an unbounded variable ⇒ the state space is infinite
- Timeout signals can be added to the queues at any point of the time slice, and be consumed at some point later ⇒ blowing up the state space due to the sheer amount of possible interleaving sequences of events.
- Keeping a timeout signal in a process queue increases the length of the state vector.

Some History
(Translating SDL into Promela)

Abstracted timers:

G. Holzmann and J. Patti,
Validating SDL Specifications: an Experiment,
PSTV'89.

Timers modelled as processes sending timeout signals when the system is blocked. As the result some behaviour is lost.

H. Tuominen
Embedding a Dialect of SDL in PROMELA,
6th Int. SPIN Workshop, LNCS 1680

Timers can only expire in the order they are set, no matter what values they are set to.

Abstracted and concrete timers:

Bosnacki, Dams, Holenderski, Sidorova,
Verifying SDL in Spin,
TACAS 2000, LNCS 1785.

DT Spin semantics is used to represent SDL timers. Abstracted version: timers, being set, can expire at an arbitrary moment.

Timer Transformation

F:

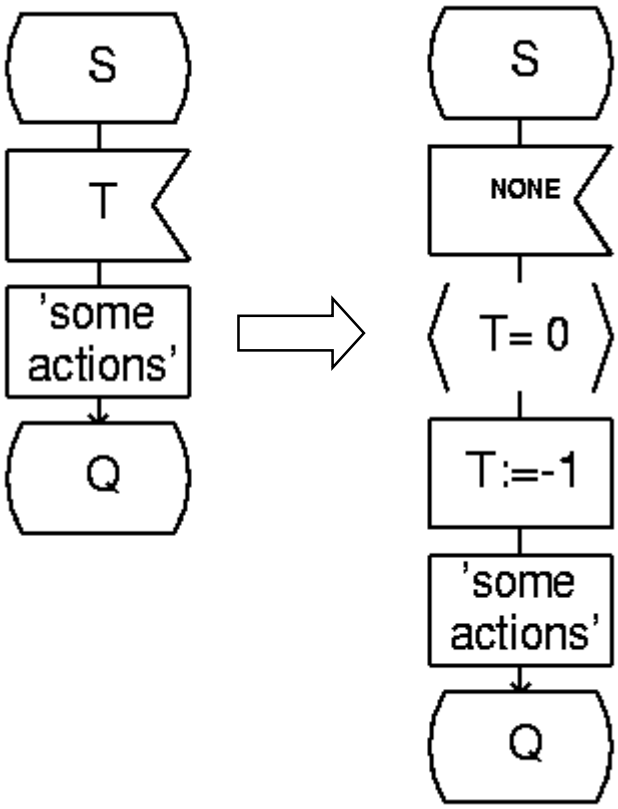
$$off \rightarrow -1$$

$$\begin{aligned} on(NOW) &\rightarrow 0 \\ on(NOW+1) &\rightarrow 1 \\ &\cdots \\ on(NOW+k) &\rightarrow k \\ &\cdots \end{aligned}$$

$$SET(NOW+k,T) \rightarrow T:= k$$

$$RESET(T) \rightarrow T:= -1$$

Timer Transformation
(continued)



Time progress:

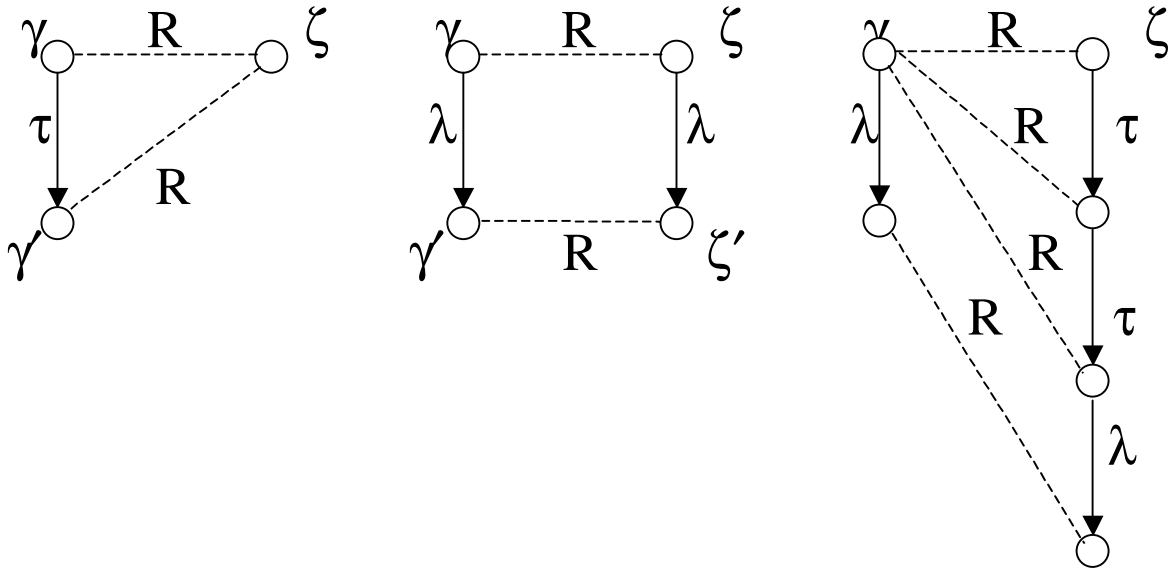
If the system is blocked then
TICK: NOW → NOW + 1

is transformed to

If the system is blocked then
for every timer T : (T>0 ⇒ T:= T – 1)

Is the Behaviour Preserved?

Simulation relation:



γRζ iff

- the valuations of variables are equal;
- projections of queues are equal (timeouts are projected out);
- for timers, either $F(t_\gamma) = t_\zeta$, or $t_\gamma = \mathit{off}$ and its timeout signal is in the process queue, and $t_\zeta = 0$.

Is the Behaviour Preserved?
(continued)

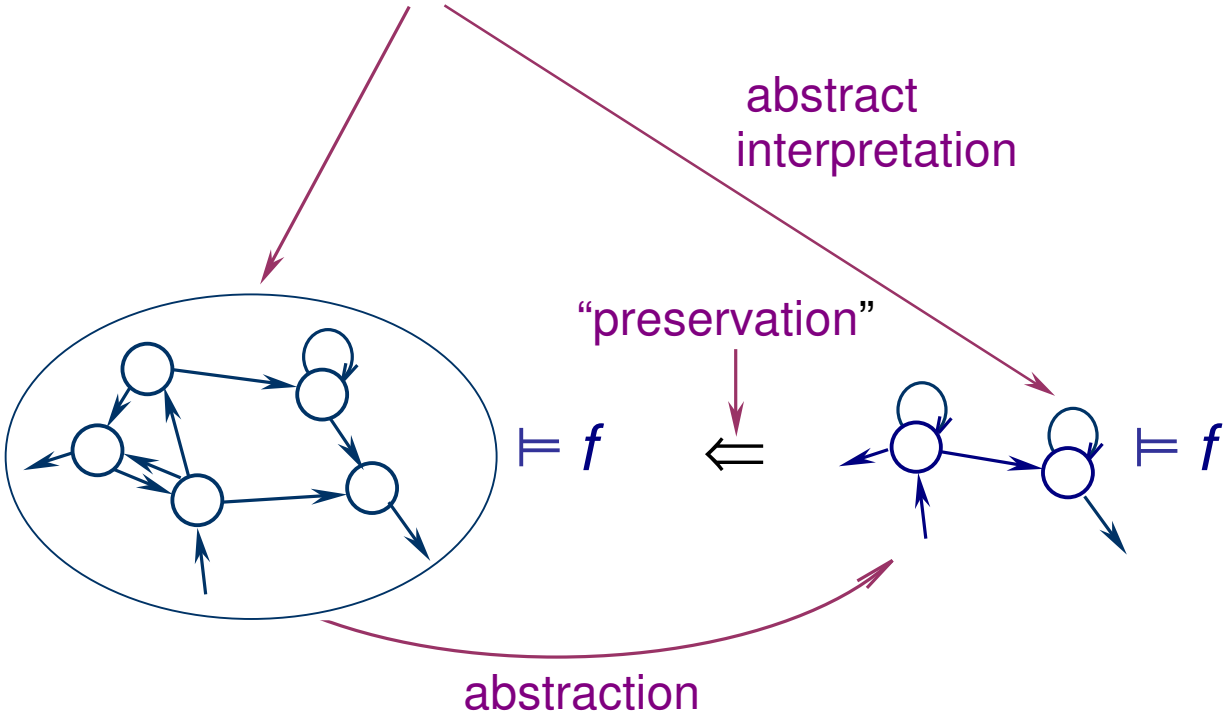
R and R^{-1} are the simulation relations.

For all formulas φ from next-free LTL,
φ holds for the original system iff
it holds for the transformed one.

Abstraction Techniques

Abstraction:

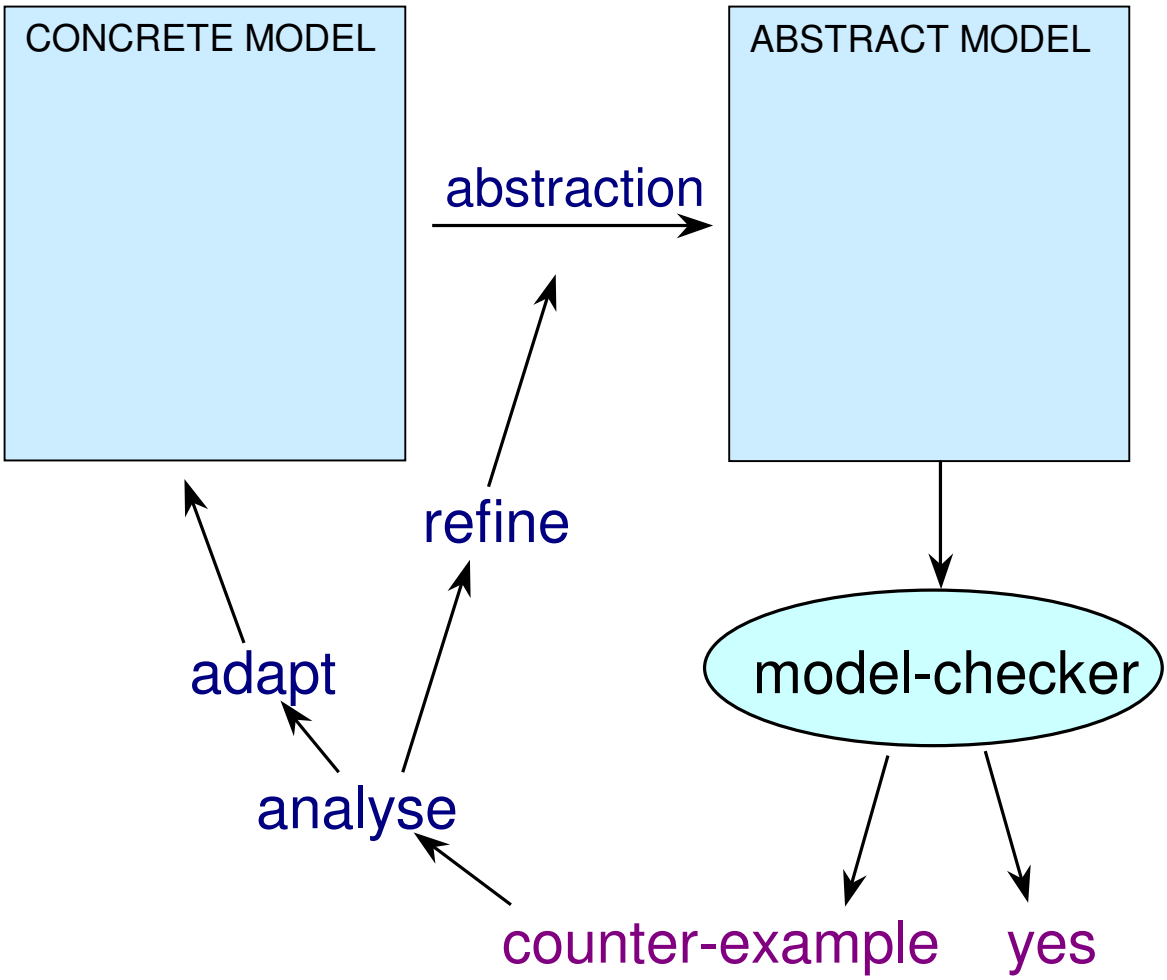
Does system $\mathcal{S}$ satisfy property f ?

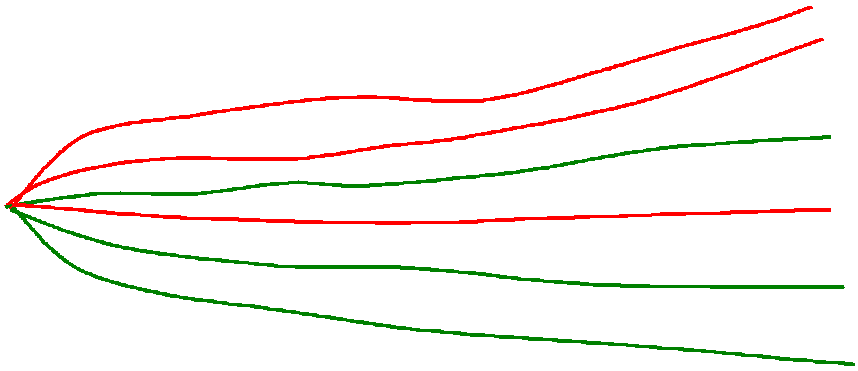


Safe abstractions:

Every property checked to be true on the abstract model holds for the concrete one as well

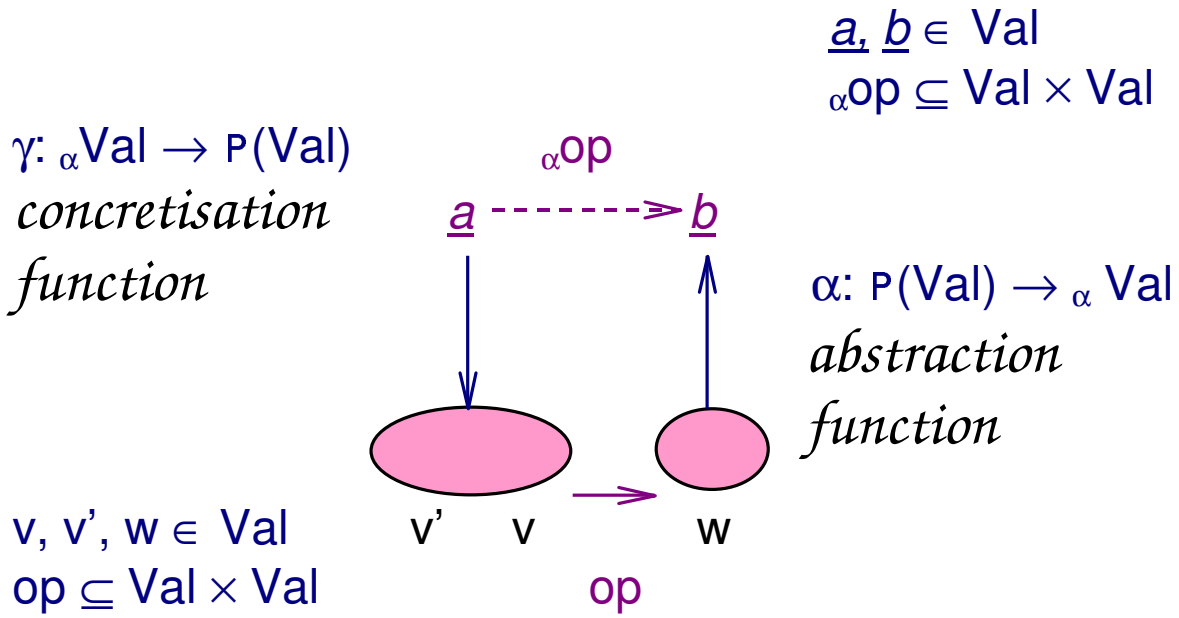
Applying Safe Abstractions





Timer Abstractions

Idea (like normal data abstractions):
Replace timer values by descriptions (abstract values) and "mimic" operations. Then every universal property checked to be true on the abstract system holds for the concrete system as well.



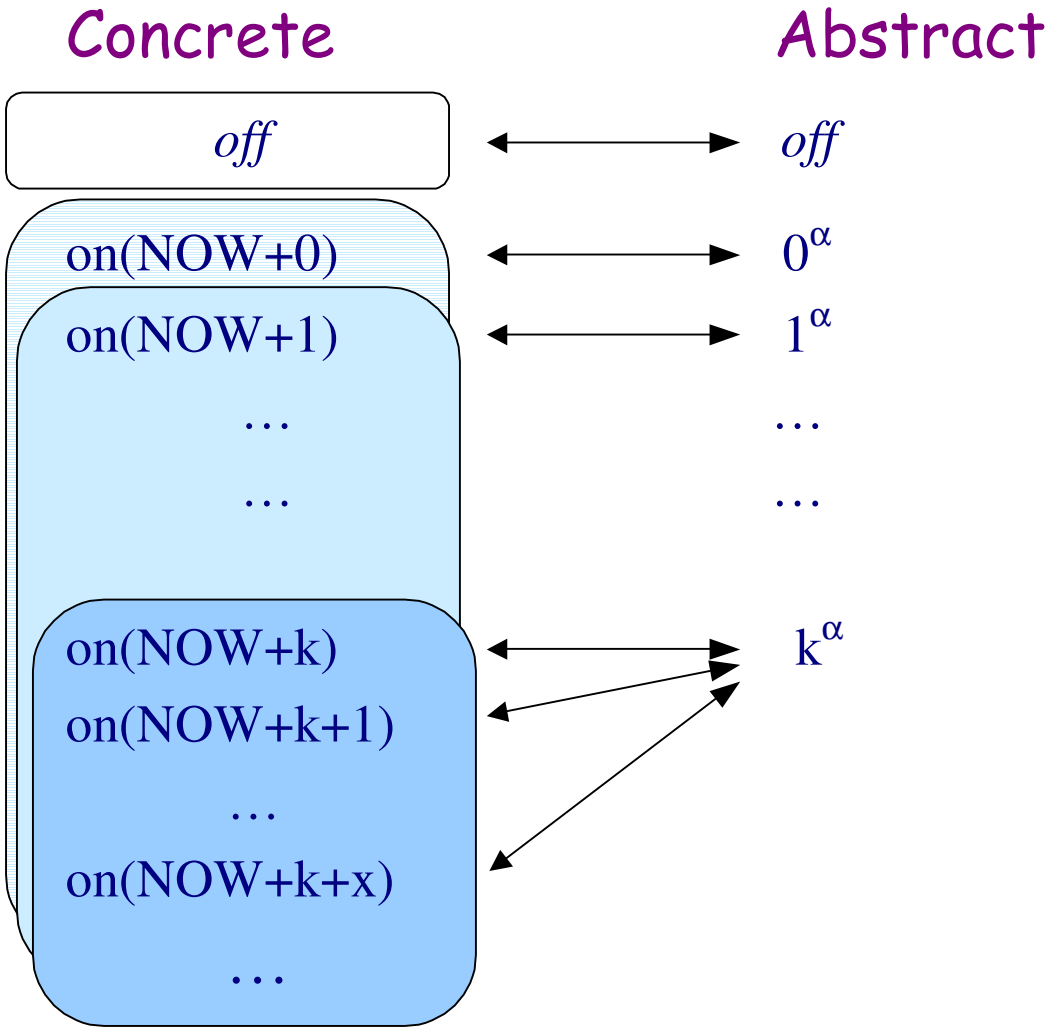
(α, γ) Galois Insertion

[Cousot & Cousot 77
Clarke, Grumberg & Long 92
Graf & Loiseaux 92
Dams, Gerth & Grumberg 94]

Timer Abstraction: How?

(Finite) Abstract data domain:
$off, 0^{\alpha}, 1^{\alpha}, \ldots, k^{\alpha}$.

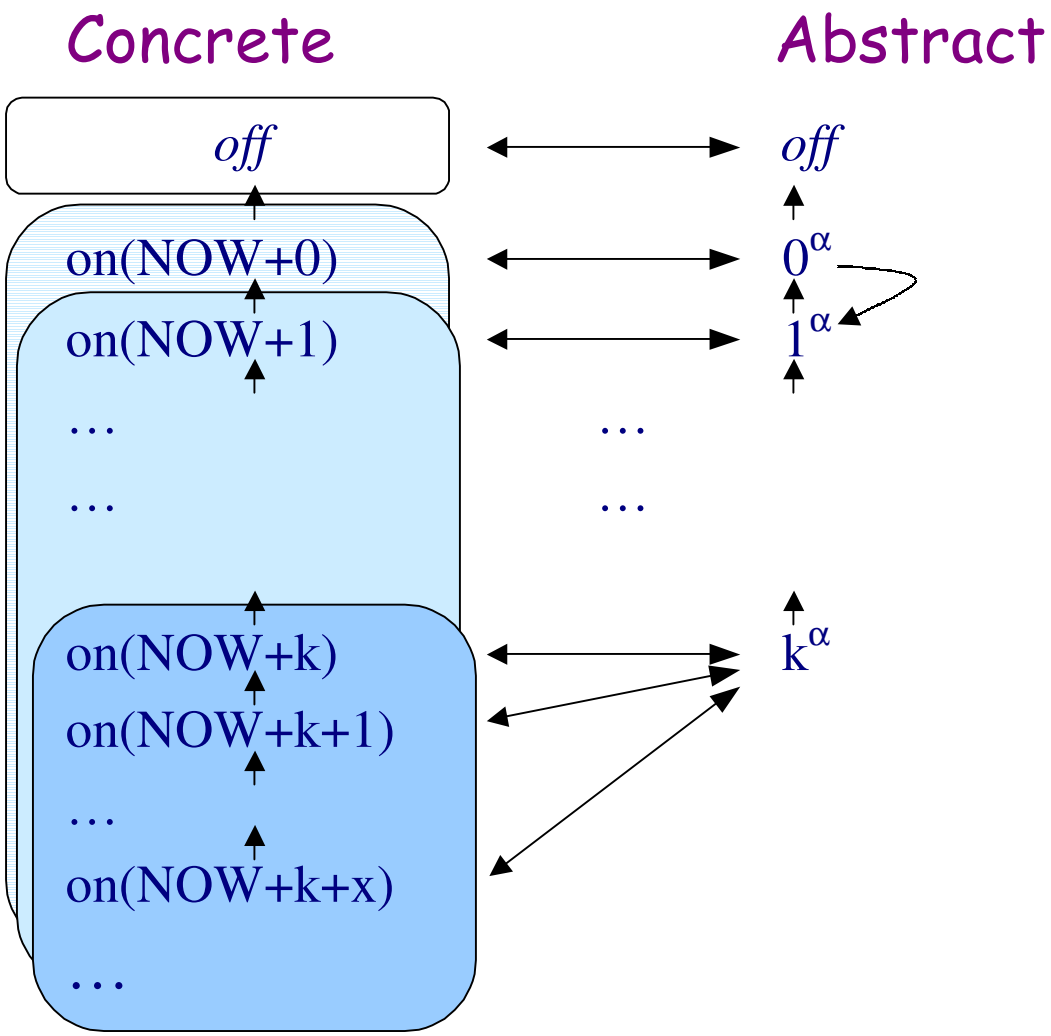
$\gamma(m^{\alpha}) = \{on(NOW+i) \mid i \geq m\}$.



$0^{\alpha} \succeq 1^{\alpha} \succeq 2^{\alpha} \succeq \ldots \succeq k^{\alpha}$.

Is the Abstraction Safe?

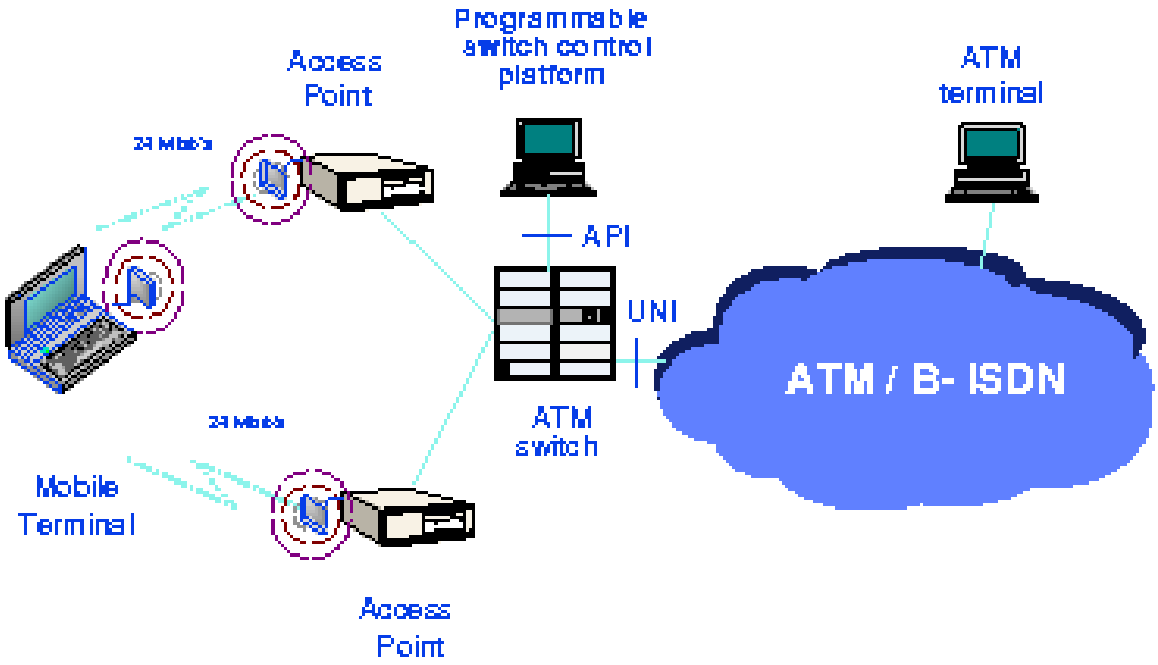
Timer expiration:



$$\forall x : \alpha(f_{\text{conc}}(x)) \preceq f_{\text{abs}}(\alpha(x))$$

The abstraction is safe !

Application



Mascara: a medium-access layer for wireless ATM communication in local area networks, developed within the WAND project (Intracom).

3-values abstraction was the most effective:

$off \rightarrow off$

$0, 1, 2, \ldots \rightarrow 0$

$1, 2, 3, \ldots \rightarrow 1$

Conclusions

- The proposed transformation is a simple and cheap zero-phase of the verification process.
- The new time semantics gives a better understanding of the SDL semantics.
- Timer abstractions are efficient in practice.
- "Cheap" techniques can be more efficient than "expensive" ones!