Scheduled Migration in Distributed Systems

Gabriel Ciobanu

Romanian Academy, Institute of Computer Science

700505 Iasi, Romania

http://www.info.uaic.ro/~gabriel

FROM 2017

Bucharest, 5th July 2017

Basic Features of TiMo

- Modelling distributed systems with time-related aspects
- Simple process algebra:

  locations+ mobility + interaction + timers
- Local interaction (communication) and local clocks
- Communication of locations between processes
- Migration is no-urgent, modelling network delays
- Interaction (communication) is not delayed
- Discrete time semantics + maximal concurrency

Example: simple e-shops

- In this scenario, we have a client process which initially resides in the *home* location, and wants to find an address of an e-shop where different kinds of electronic items (e-items) can be purchased.
- To find out the address of a suitable e-shop, the client, within 2 time units, moves to the location *info* in order to acquire the relevant address.
- The location *info* contains a broker who knows all about the availability of the e-shops stocking the desired e-item. In the first 5 time units the right e-shop is the one at the location *shopA* , and after that for 7 time units that at location *shopB* .

Example: simple e-shops

- Since interaction happens within the same location, it is necessary for the client process to move to the broker location in order to find out about the e-item.
- The timers can define a coordination in time and space of the client, and take care of the relative time of interaction of the processes residing at the same location.

Example: simple e-shops

- The specification of the running example which captures the essential features of the scenario described previously can then be written down in the following way:

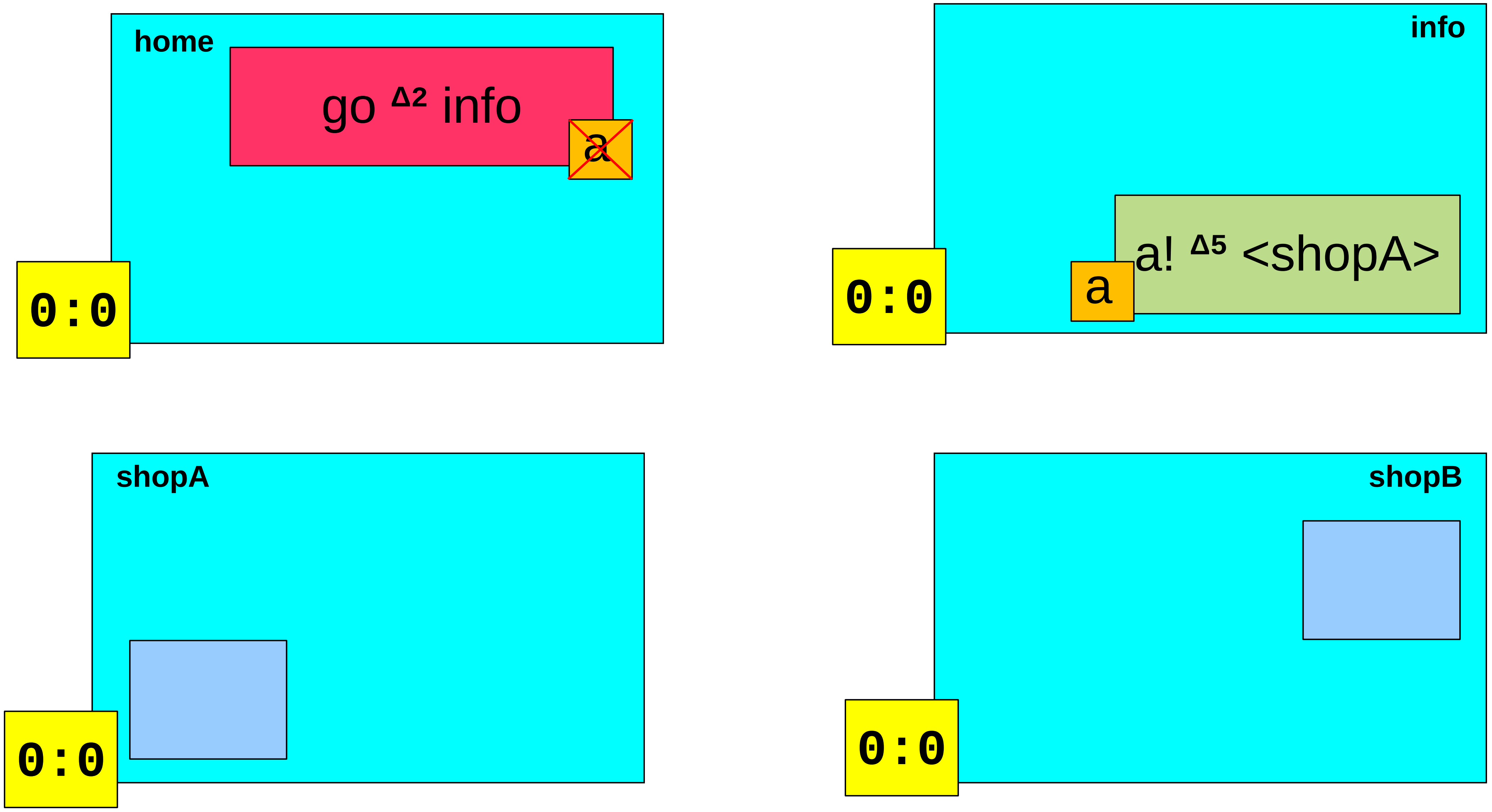
$$SES = home[Client] \mid info[Broker] \mid shopA[\,] \mid shopB[\sim]$$

- where:

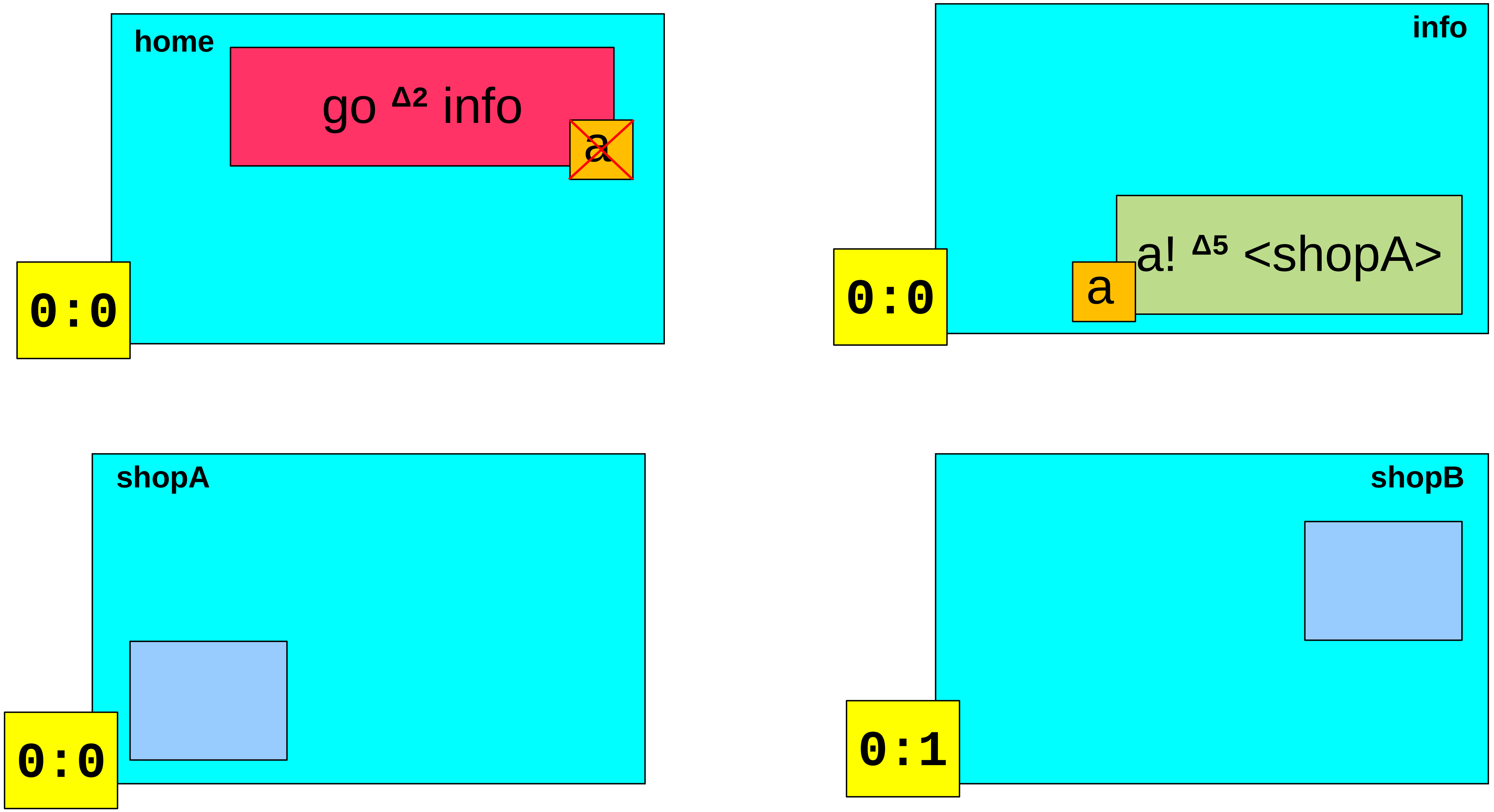
$$Client = go^{\Delta 2}\ info\ .\ (a^{\Delta 2}\ ?\ (shop)\ then\ go^{\Delta 0}\ shop\ else\ go^{\Delta 0}\ home\ )$$

$$Broker = a^{\Delta 5}\ !\ \langle shopA \rangle\ then\ 0\ else\ a^{\Delta 7}\ !\ \langle shopB \rangle$$

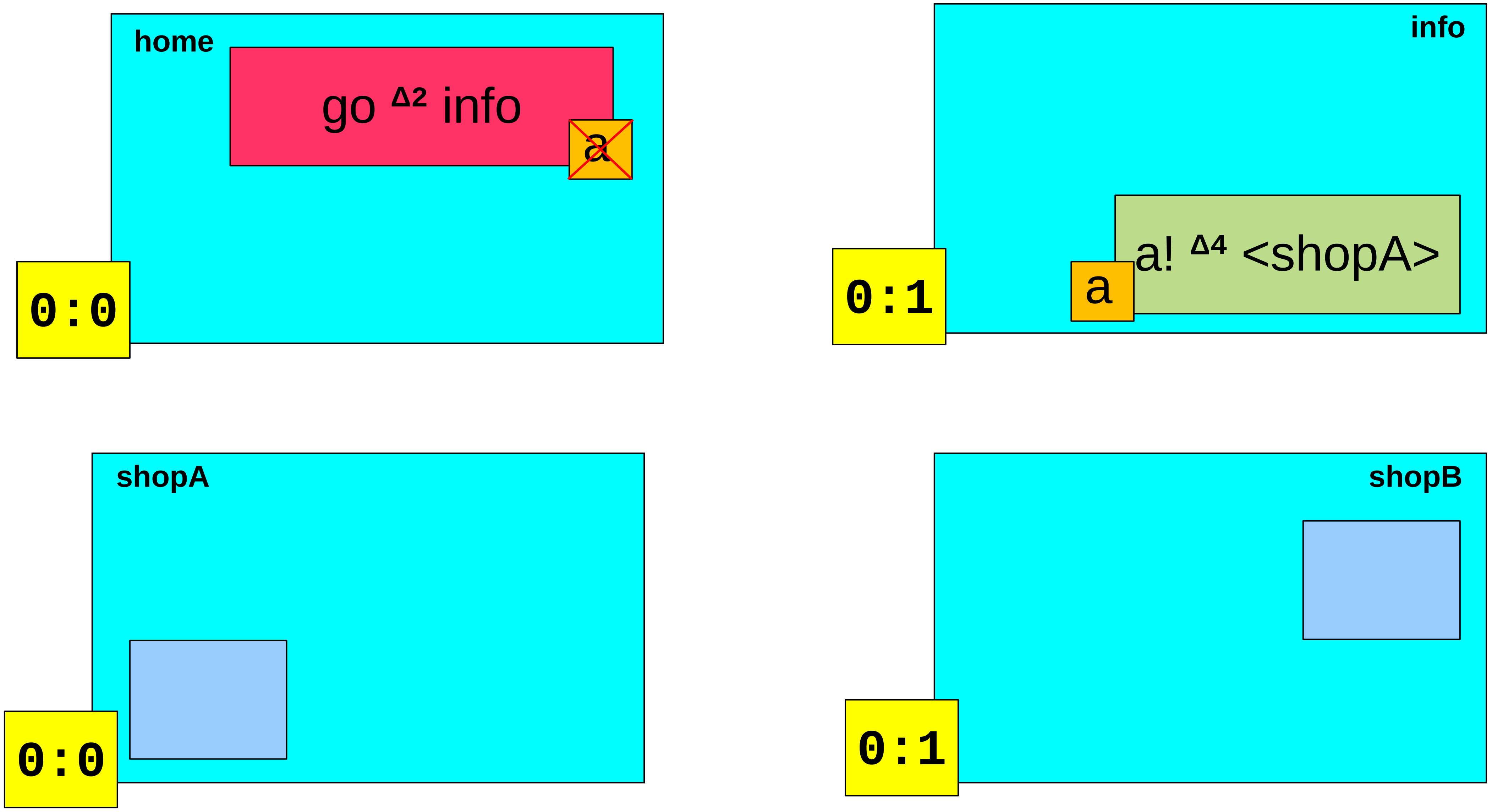
Example: simple e-shops



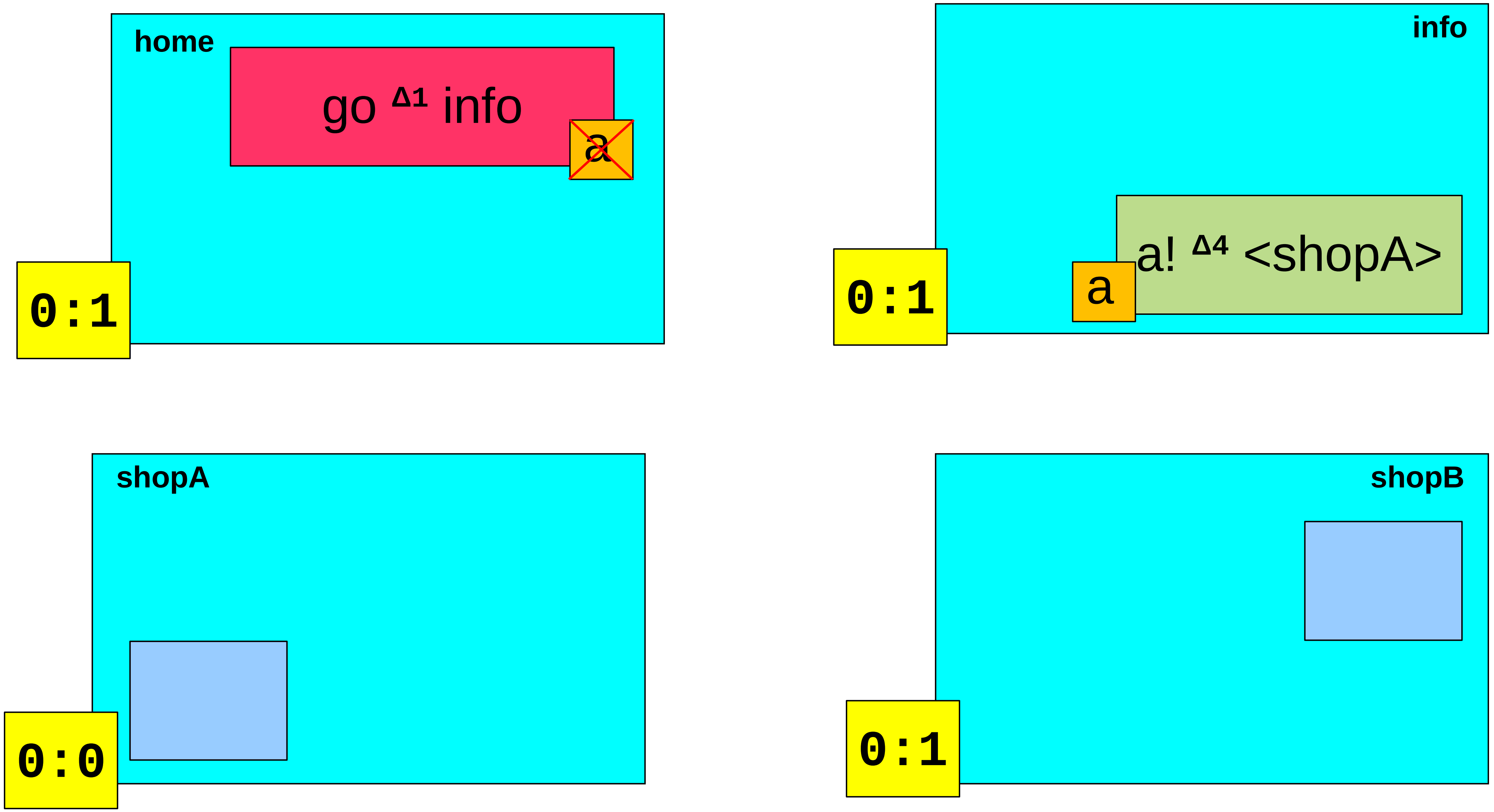
Example: simple e-shops



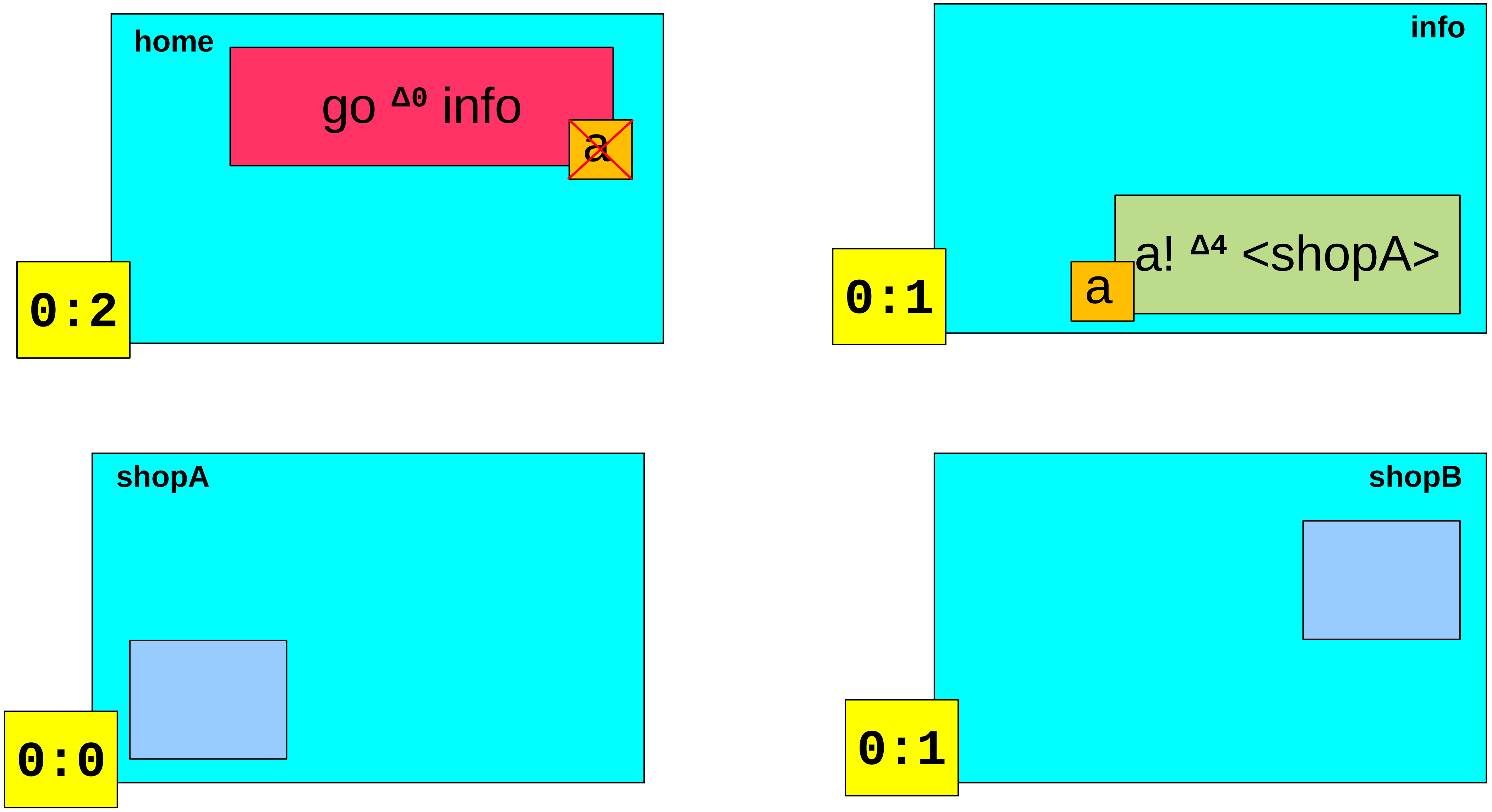
Example: simple e-shops



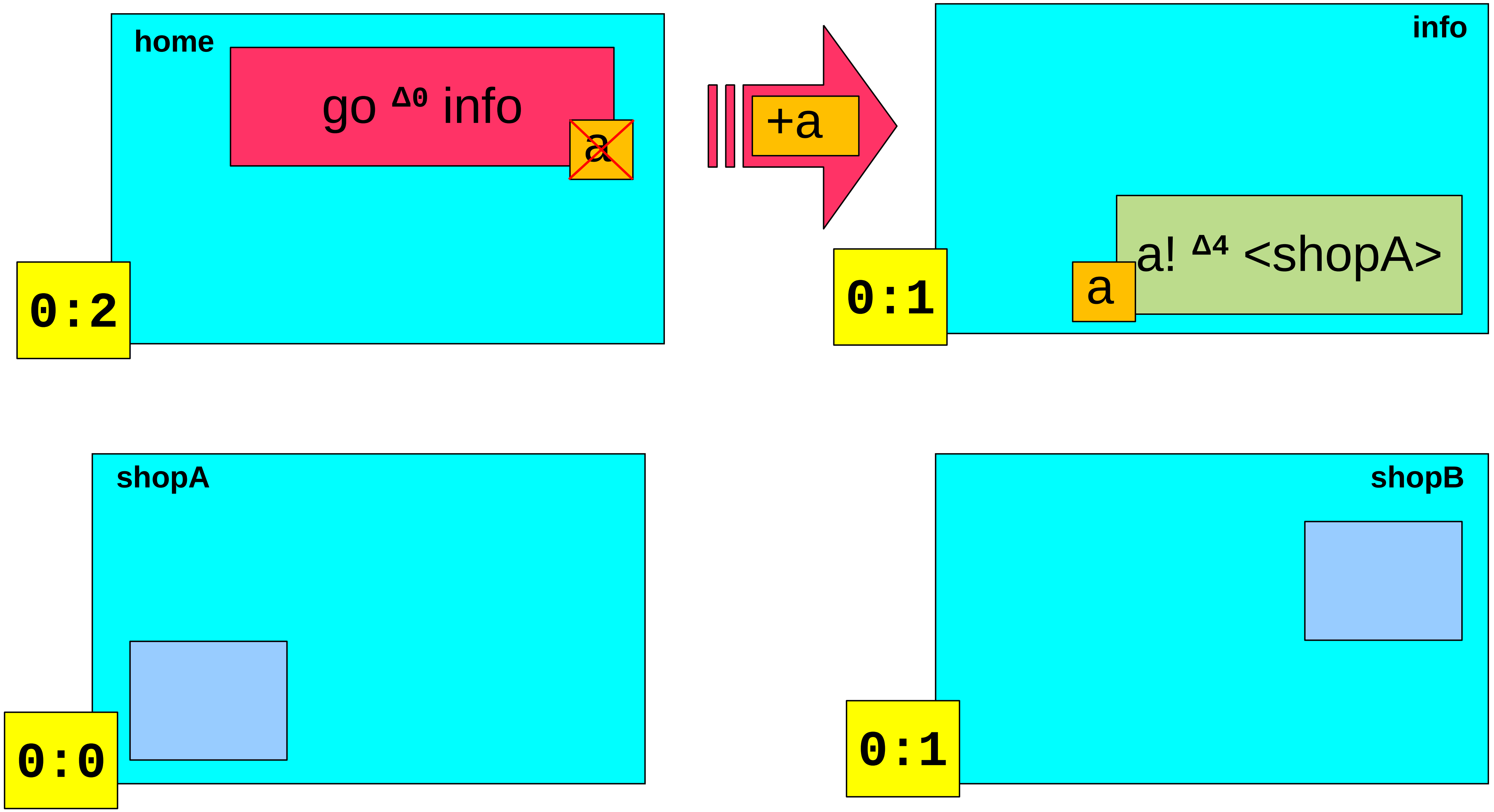
Example: simple e-shops



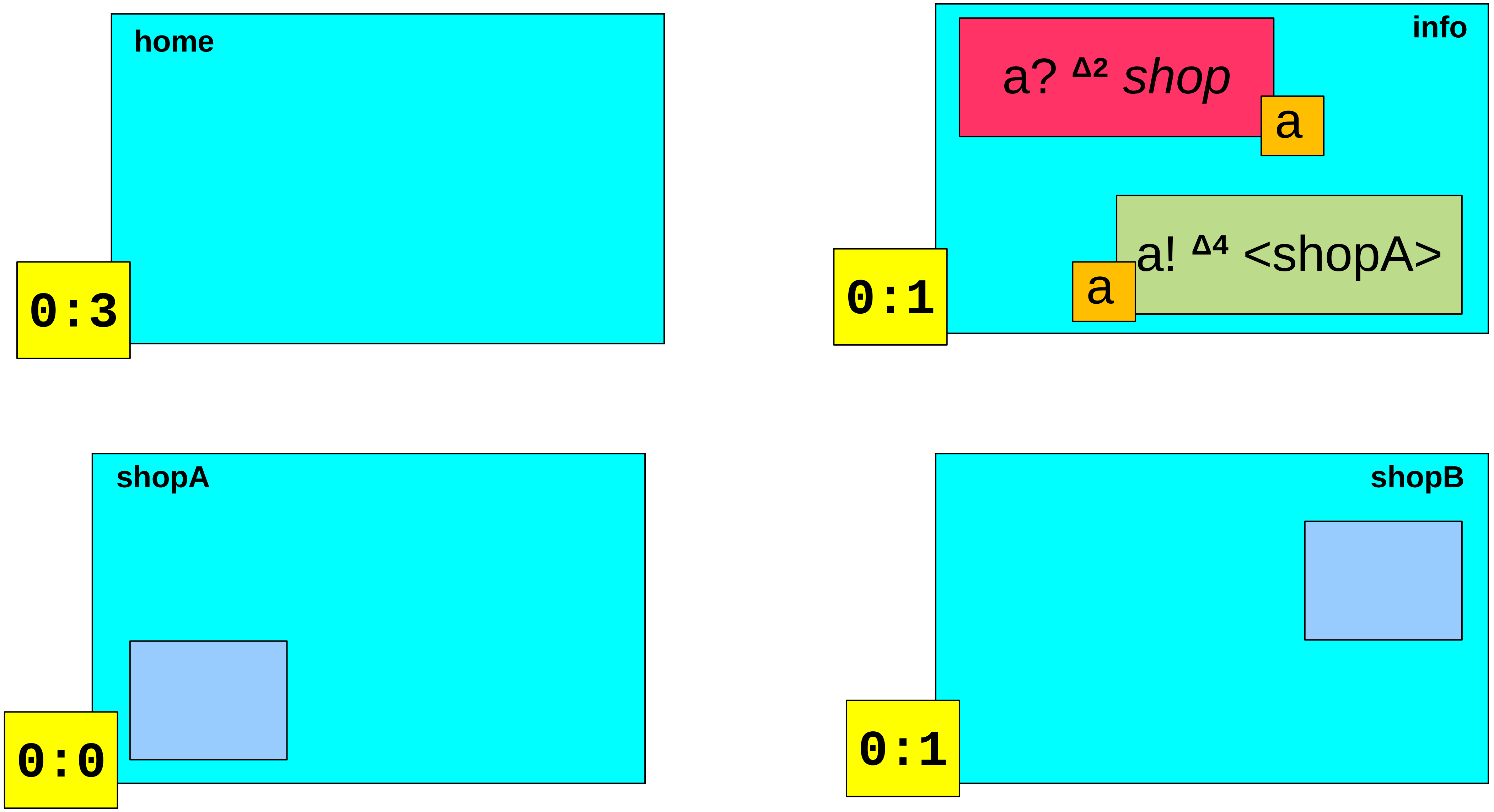
Example: simple e-shops



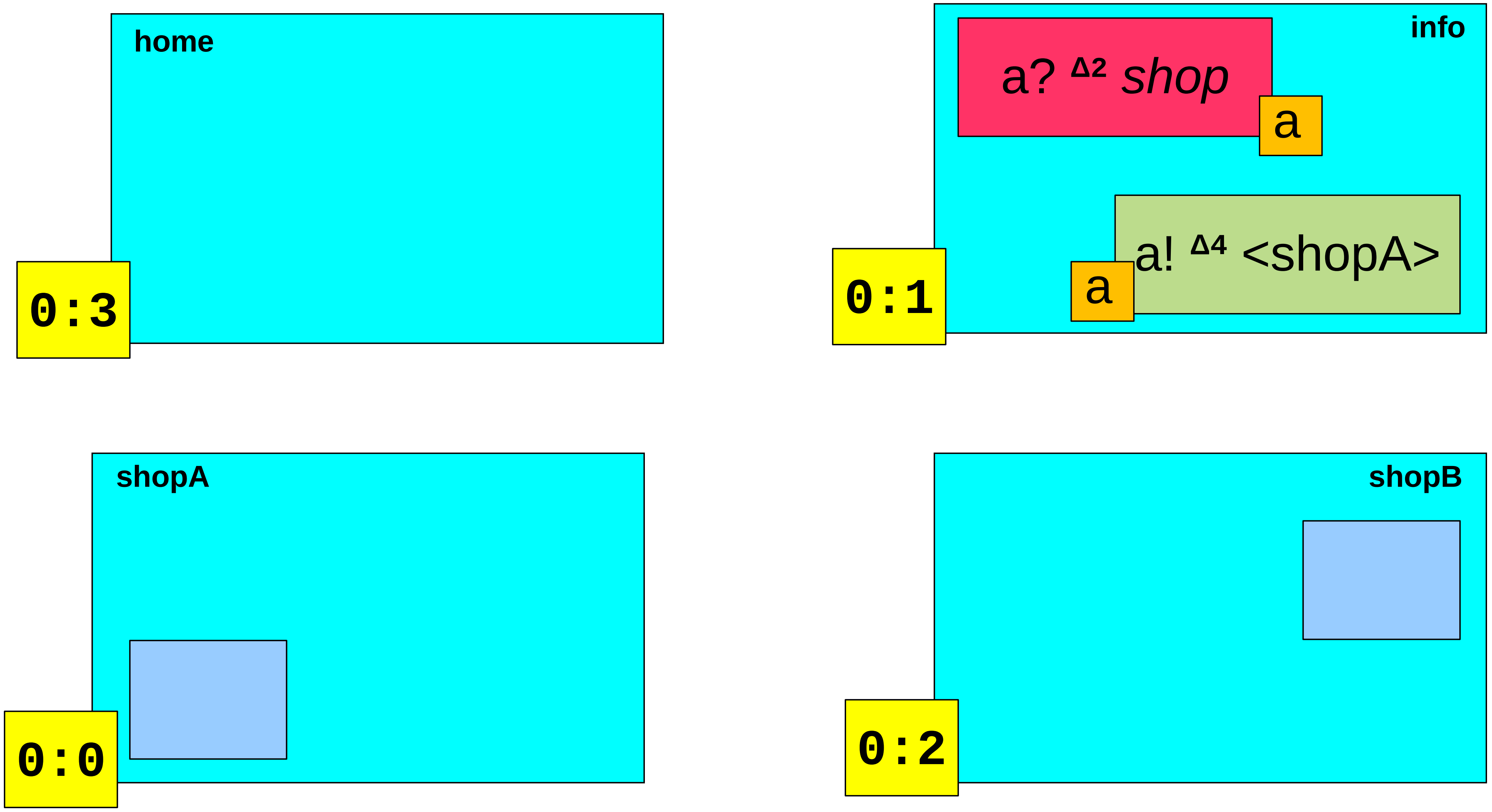
Example: simple e-shops



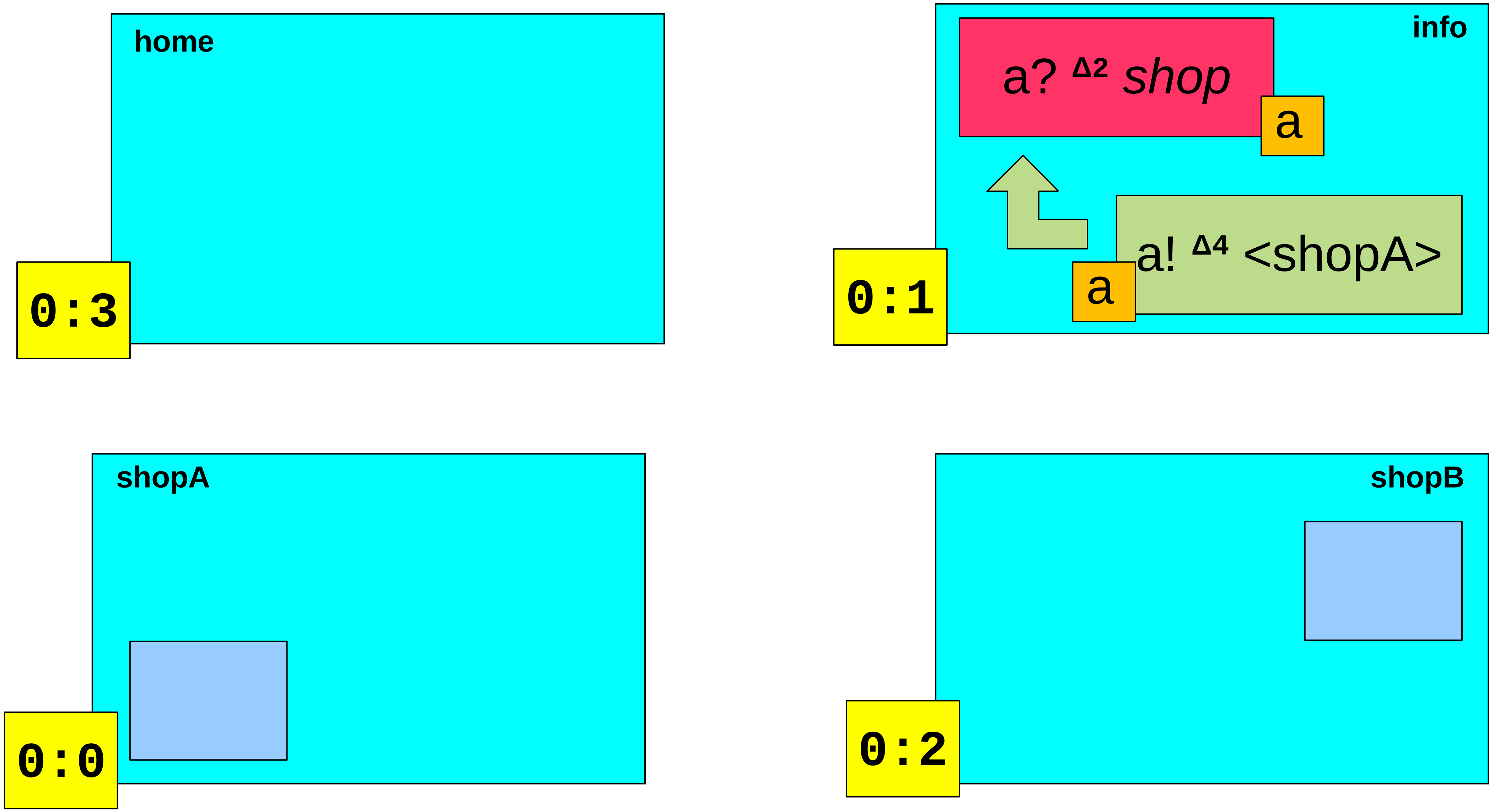
Example: simple e-shops



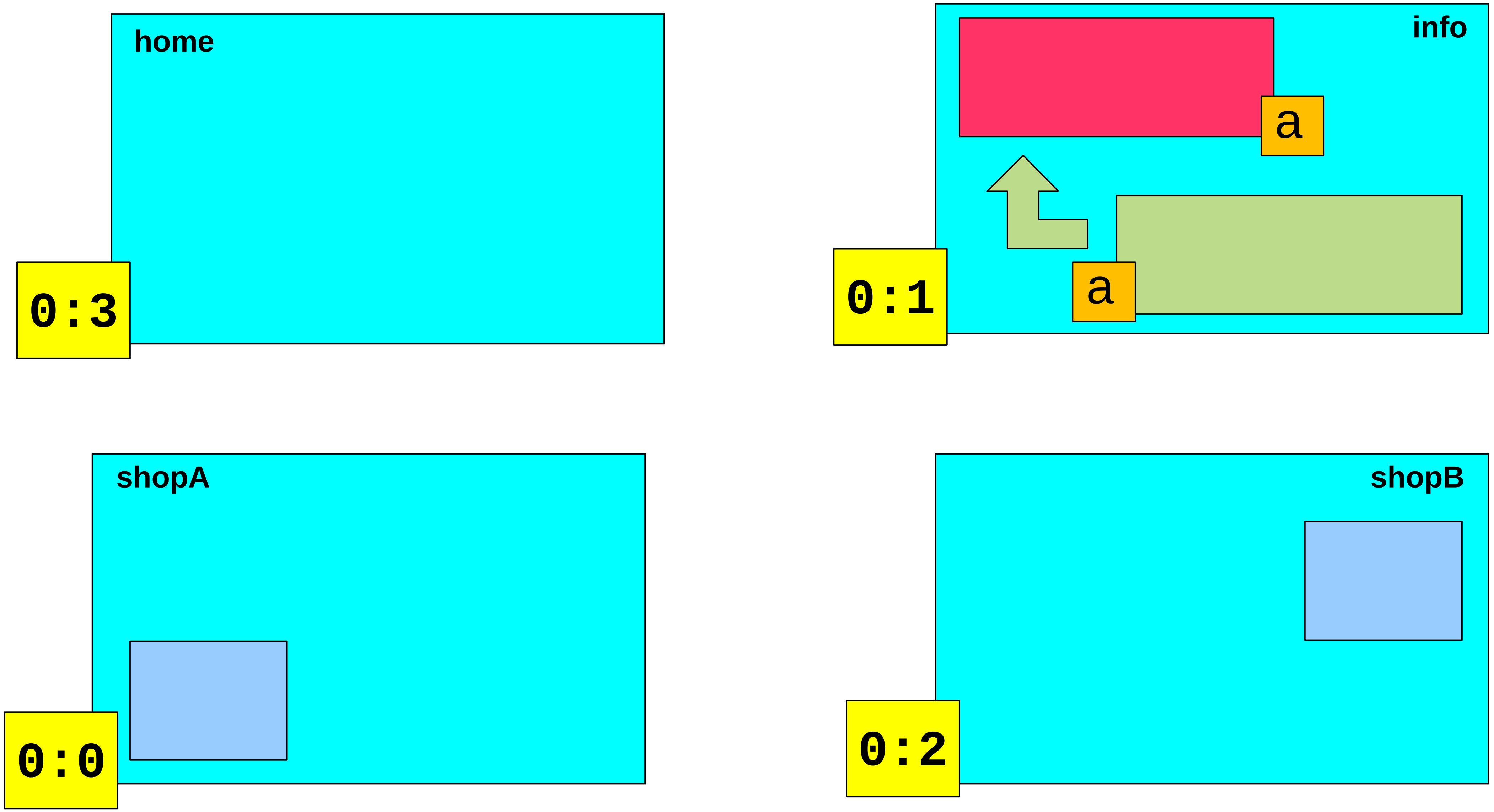
Example: simple e-shops



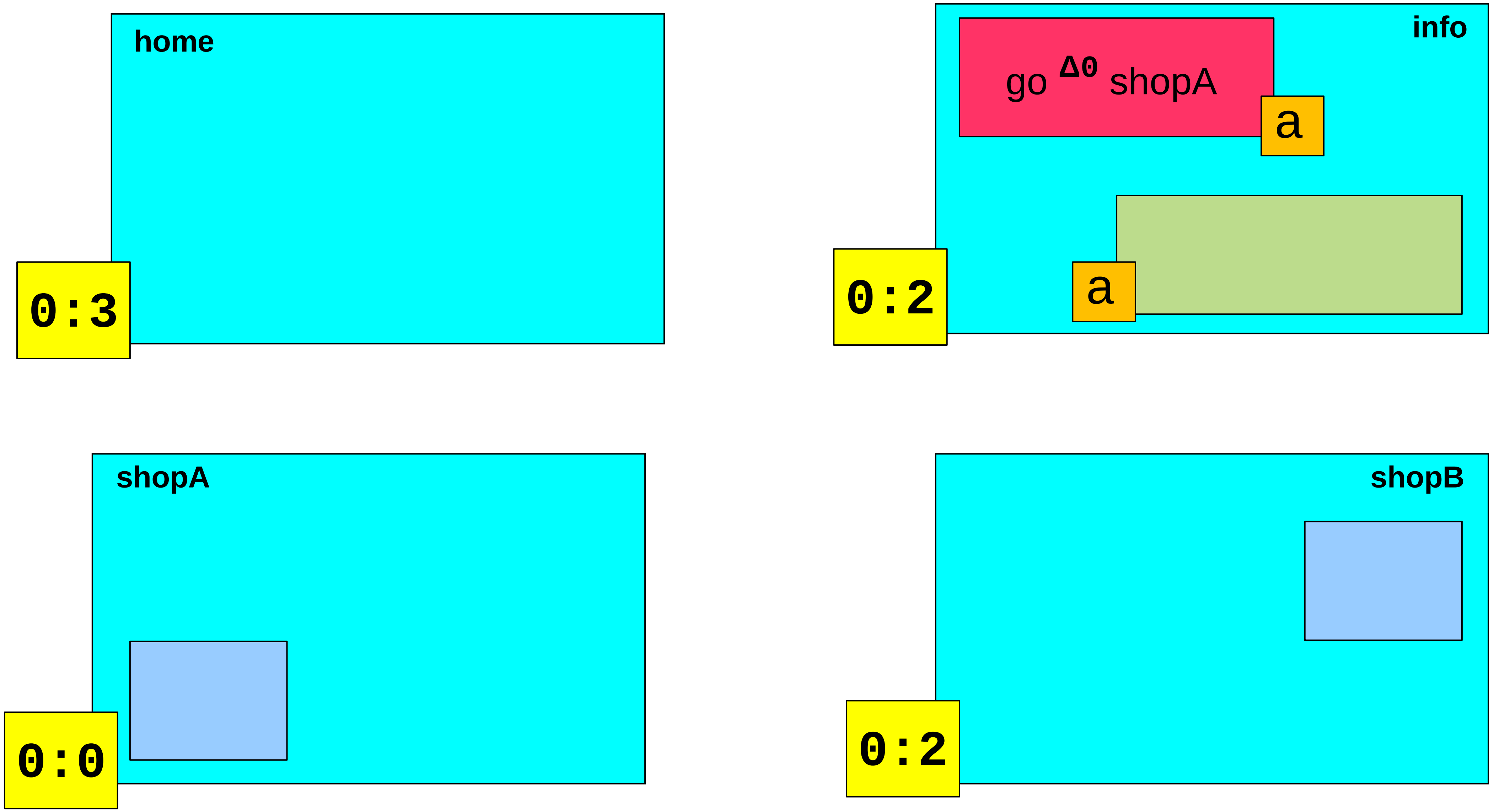
Example: simple e-shops



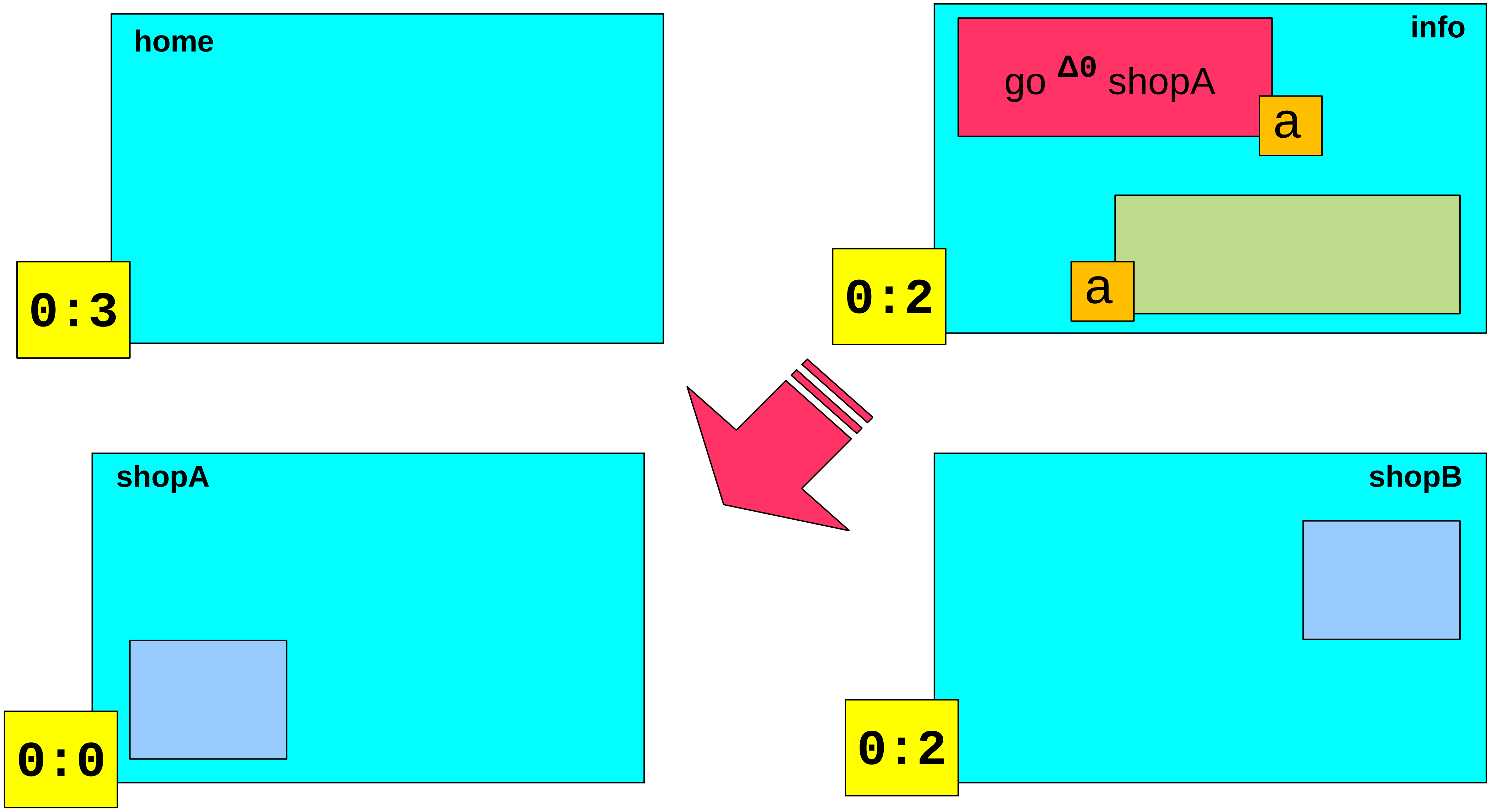
Example: simple e-shops



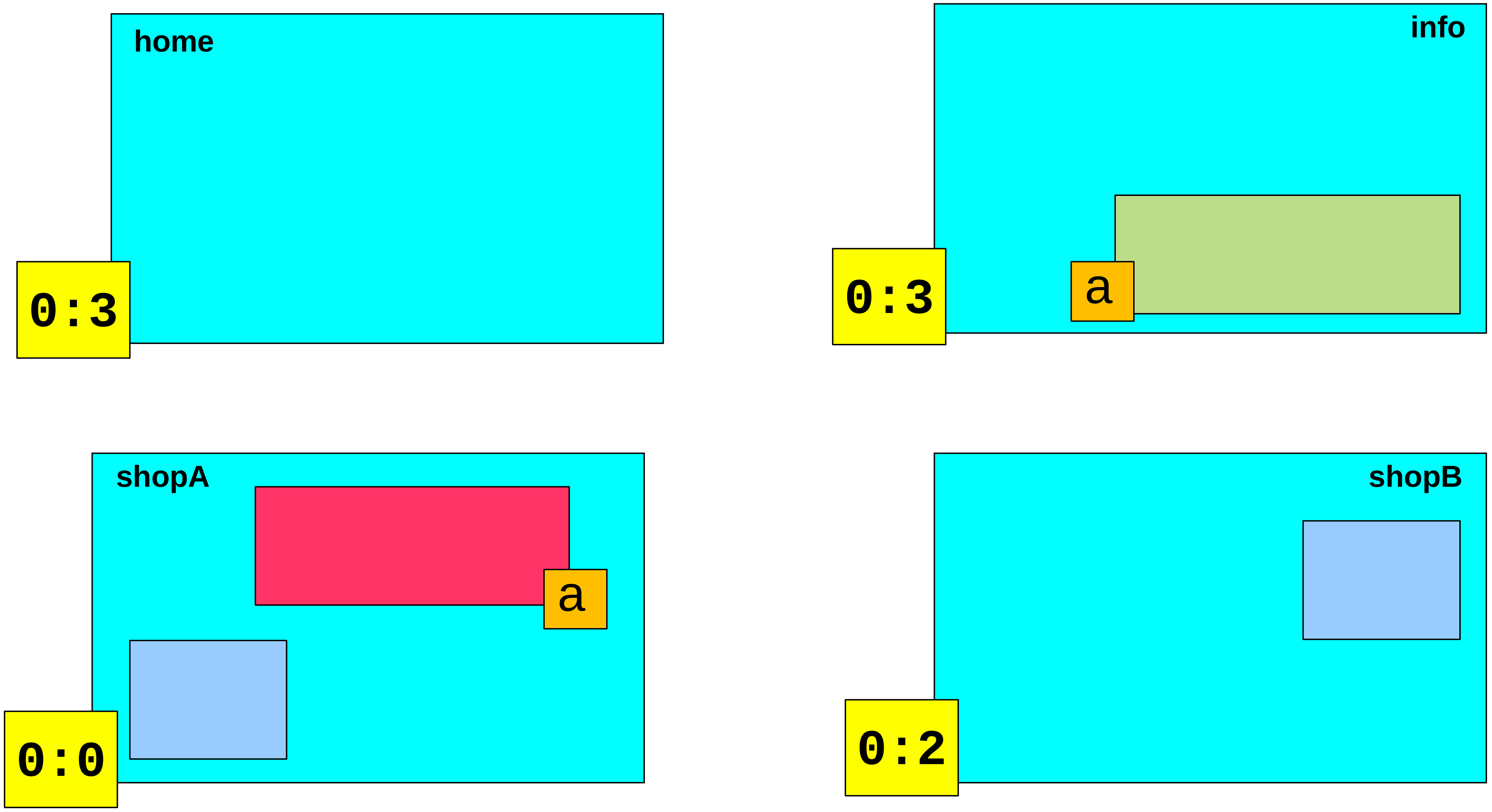
Example: simple e-shops



Example: simple e-shops



Example: simple e-shops



TiMo Syntax

a b c ...	channels (names)
k m ...	locations
$a^{\Delta t}$?(u) then P else Q	input
$a^{\Delta t}$!<v> then P else Q	output
$go^{\Delta t}$ v then P	mobility
stop	do nothing

TiMo Syntax

P | Q	parallel
$\circledS$ P	stalled process
id(u1, u2, ...)	process identifier
N | M	network

Γ Γ’ ...	sets of access permissions
P: Γ	process with access permissions
k [PP]	located process with permissions

Output / Input - intuition

$a^{\Delta t}$!<v> then P else Q

can send v over channel a if put<a@k> is present for t time units and continue as P; if unsuccessful, continues as Q

$a^{\Delta t}$?(u) then P else Q

can input some value if get<a@k> is present for t time units, and substitute it for variable u within its body (u is bound within P, but not within Q); then continues as P; if unsuccessful, continues as Q

Migration - intuition

go$^{\Delta t}$ u then P

waits for t time units before migrating and continuing as P

u can be assigned value dynamically through communication with other processes

TiMo Operational Semantics

(MOVE)

$$k\,[\,go^{\Delta 0}\ m\ \text{then}\ P\,] \xrightarrow{k:m} m\,[\circledS P]$$

(WAIT)

$$k\,[\,go^{\Delta t}\ m\ \text{then}\ P] \xrightarrow{k} k\,[\circledS\ go^{\Delta t\text{-}1}\ m\ \text{then}\ P]$$

(COM)

$$k\,[\,a!^{\Delta t}\langle v\rangle\ \text{then}\ P\ \text{else}\ Q\ \mid\ a?^{\Delta t'}(u)\ \text{then}\ P'\ \text{else}\ Q'\ ]$$

$$\xrightarrow{a(v)@k} k\,[\circledS P\ \mid \circledS\{v/u\}P'\ ]$$

TiMo Operational Semantics Rule (TIME)

(TIME)

$$N \xrightarrow{\surd k} \varphi_k(N)$$

- applicable if the other action rules cannot be applied at location k (maximal progress at a location k)
- φ_k (N) updates timer values, selects continuations, and removes $\circledS$ ’s

Execution (operational semantics)

$k\,[\,go^{\Delta 2}\,m\,.\,a?^{\Delta 1}(u).P\,]\;|\;m\,[\,a!^{\Delta 4}\langle h\rangle\,]$ $\xrightarrow{\surd m}$ $\boxed{\text{(TIME)}}$

$k\,[\,go^{\Delta 2}\,m\,.\,a?^{\Delta 1}(u).P\,]\;|\;m\,[\,a!^{\Delta 3}\langle h\rangle\,]$ $\xrightarrow{@k}$ $\boxed{\text{(WAIT)}}$

$k\,[\,\circledS o^{\Delta 1}\,m\,.\,a?^{\Delta 1}(u).P\,]\;|\;m\,[\,a!^{\Delta 3}\langle h\rangle\,]$ $\xrightarrow{\surd k}$ $\boxed{\text{(TIME)}}$

$k\,[\,go^{\Delta 0}\,m\,.\,a?^{\Delta 1}(u).P\,]\;|\;m\,[\,a!^{\Delta 3}\langle h\rangle\,]$ $\xrightarrow{k:m}$ $\boxed{\text{(MOVE)}}$

$m\,[\,\circledS ?^{\Delta 1}(u).P\;|\;a!^{\Delta 3}\langle h\rangle\,]$ $\xrightarrow{\surd m}$ $\boxed{\text{(TIME)}}$

$m\,[\,a?^{\Delta 1}(u).P\;|\;a!^{\Delta 2}\langle h\rangle\,]$ $\xrightarrow{a(h)@m}$ $\boxed{\text{(COM)}}$

$m\,[\,\circledS h/u\}P\;|\circledS 0\,]$ $\xrightarrow{\surd m}$ $\boxed{\text{(TIME)}}$

$m\,[\,\{h/u\}P\;|\;0\,]$

x then P else 0 is x . P

x then 0 else 0 is x

Structural equivalence

(EQ1)

$M \mid N \equiv N \mid M$

(EQ2)

$(M \mid N) \mid N' \equiv M \mid (N \mid N')$

(EQ3)

$k [PP \mid QQ] \equiv k [PP] \mid k [QQ]$

(EQ4)

$k [P \mid Q : \Gamma] \equiv k [P: \Gamma \mid Q: \Gamma]$

They allow a finite component decomposition of networks unique up to the permutations of the components.

Operational semantics rule on structure

(EQUIV)

$$\frac{N \equiv N' \qquad N \xrightarrow{\lambda} M \qquad M \equiv M'}{N' \xrightarrow{\lambda} M'}$$

Operational semantics (network)

Cumulative effect of actions at location k:

$$N \xrightarrow{\lambda 1} \cdots \xrightarrow{\lambda n} \xrightarrow{\surd k} M$$

- if N is well formed (e.g. no Ⓢ's)
- then M is well formed

and for Ψ the multiset of all actions λi

$$N \xrightarrow{\Psi} M$$

Semantic Consistency

Assumption: $N \xrightarrow{\lambda 1} \ldots \xrightarrow{\lambda n} \xrightarrow{\surd k} M$

PROPOSITION

$n \leq$ the number of parallel components in N

(no unbounded sequence of actions without time progress)

PROPOSITION

$\lambda 1 \ldots \lambda n$ can be permuted and still reach M.

PROPOSITION

Networks reachable from a well-formed network are well-formed.

- It is proved a structural translation into equivalent Petri nets.

Structural Translation of TiMo into Petri nets

- Compositional translation from TiMo expressions to high level Petri nets with time constraints
- The resulting net is finite
- The transition system of the resulting net is strongly bisimilar to the transition system of the original TiMo expression
- Petri net representation provides model-checking techniques and tools + non-interleaving semantics model

PerTiMo: TiMo with access permissions

Security aspects expressed by access permissions

put<b@k>	to send to channel b at location k
get<b@k>	to receive from channel b at location k
Γ	set of access permissions of a migrating process
apm(k,m)(Γ)	change of access permissions of Γ when moving from k to m

Example:

$$Put^{-}_{b@k}(\Gamma) = \Gamma - \{put<b@k>\}$$

Safe access permissions

AIM:

... to verify that migrating process has sufficient access permissions to enable participation in *all potential* future communications, and never an unauthorized attempt happens during network evolutions ...

METHOD:

use judgements of the form:

$$\Gamma \vdash_k P$$

meaning that:

Γ are safe access permissions for P when it is started up at location k

Main results for PerTiMo

THEOREM 1 (soundness)

Having safe access permissions is preserved over the operational semantics rules

THEOREM 2 (safety of communications)

Processes with safe access permissions are not prevented from participating in communications with other processes

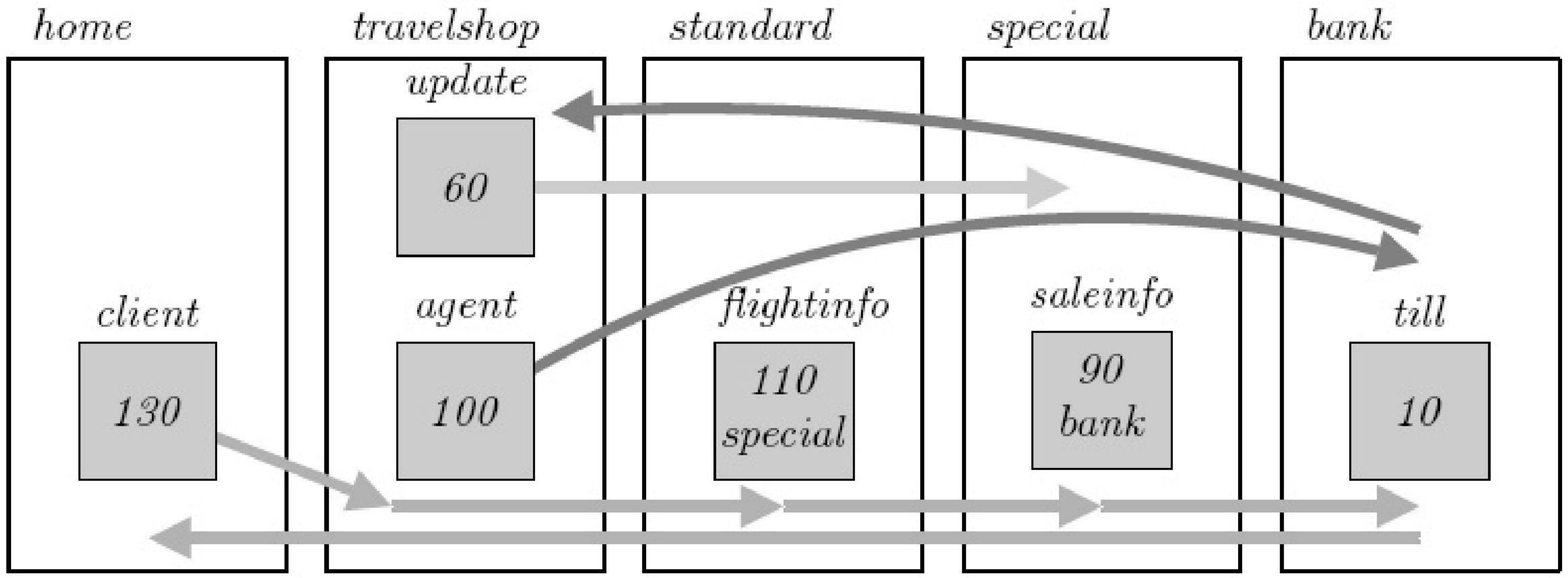
THEOREM 3 (completeness)

Processes without safe access permissions can be placed in a context which blocks a potential communication.

Travel e-Shop Example

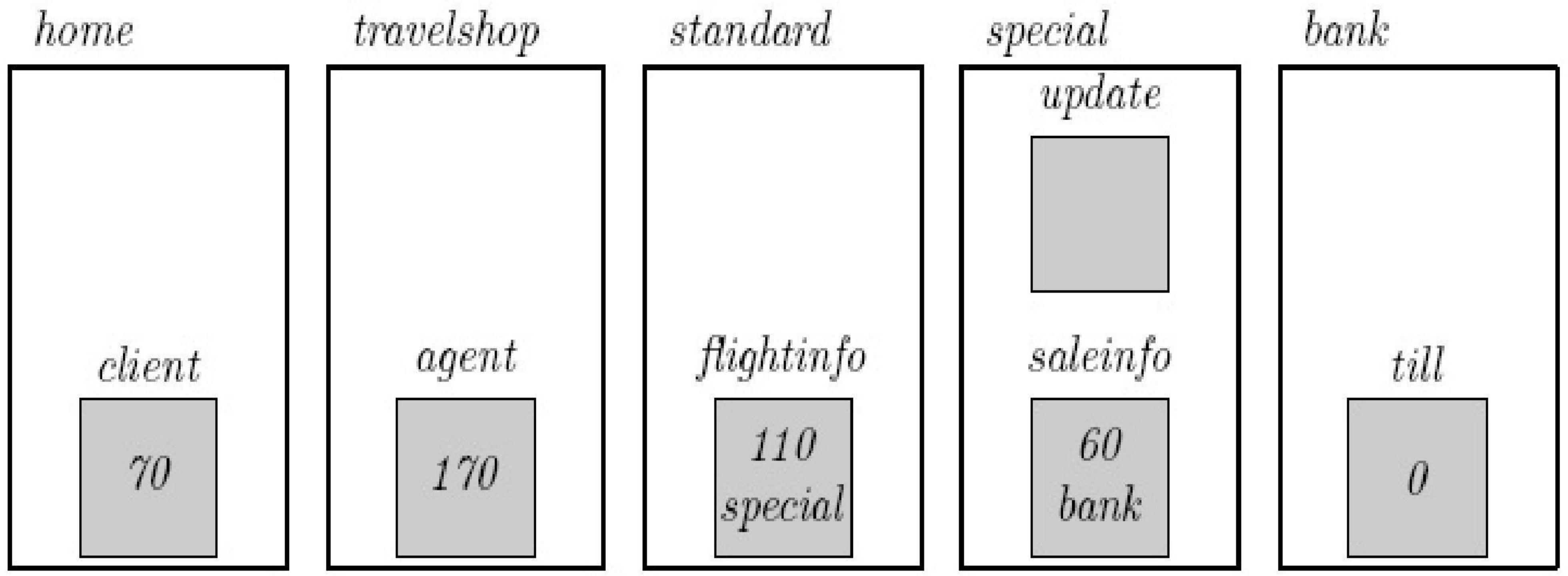
A travel shop system composed of

- six processes
- five locations



Travel e-Shop Example

A possible final network after 22 units of time is:



Property Definitions

Constraints

Bounded constraint: $expr \asymp x$ where $\asymp \in \{<, \leq\}$

Optimal constraints: max(expr) or min(expr);

Reachability property: $\mathbb{T} \leadsto K$

Process migration property: P@loc;

Bounded Liveness Property

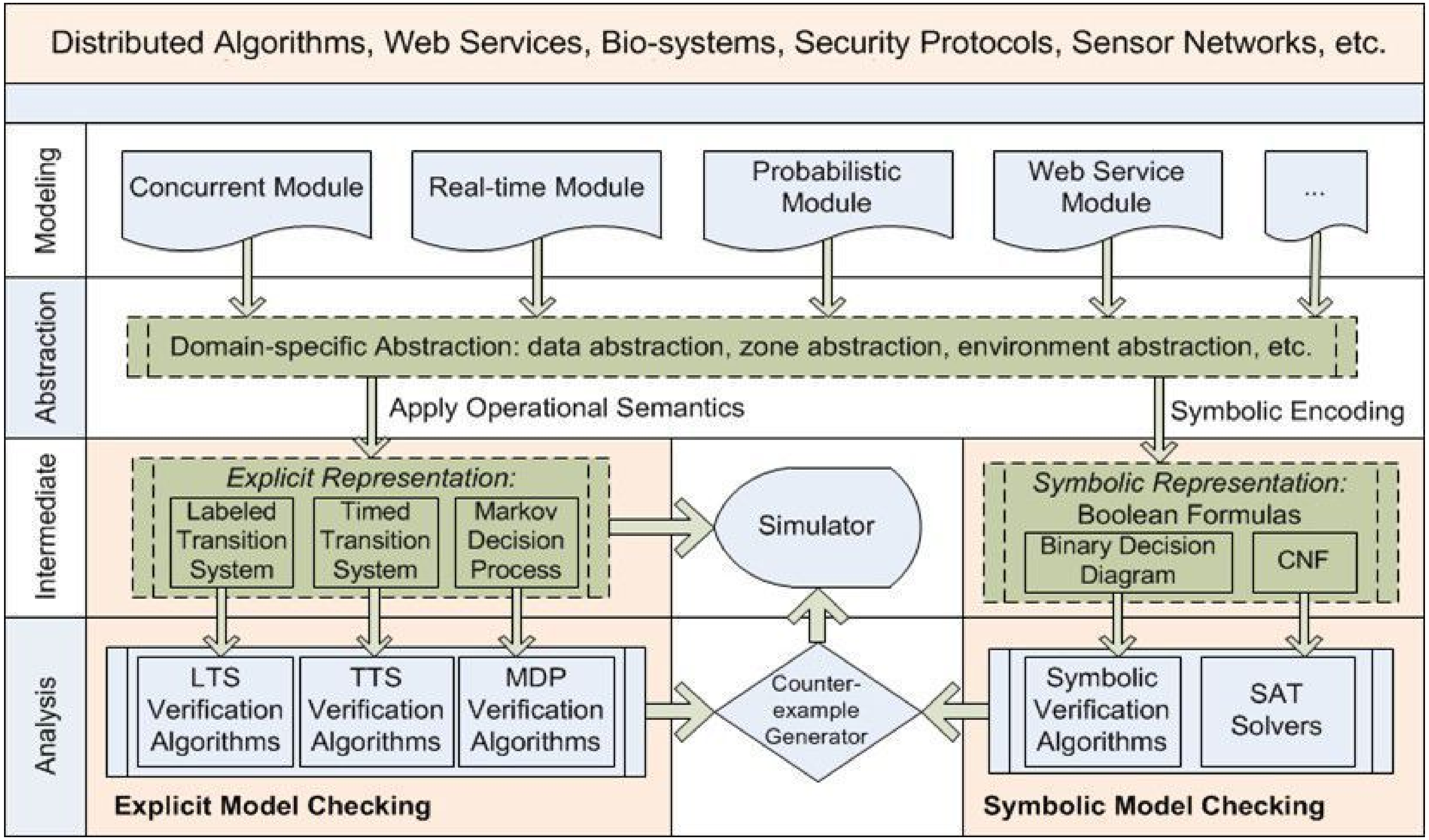
E.g., if client is able to arrive at location paying within 10 time units.

Optimized Reasoning

E.g., to find an execution path for process client to arrive at a certain location paying with the shortest time

Automatic Analysis in PAT

PAT is a extensible framework for developing domain-specific model checkers

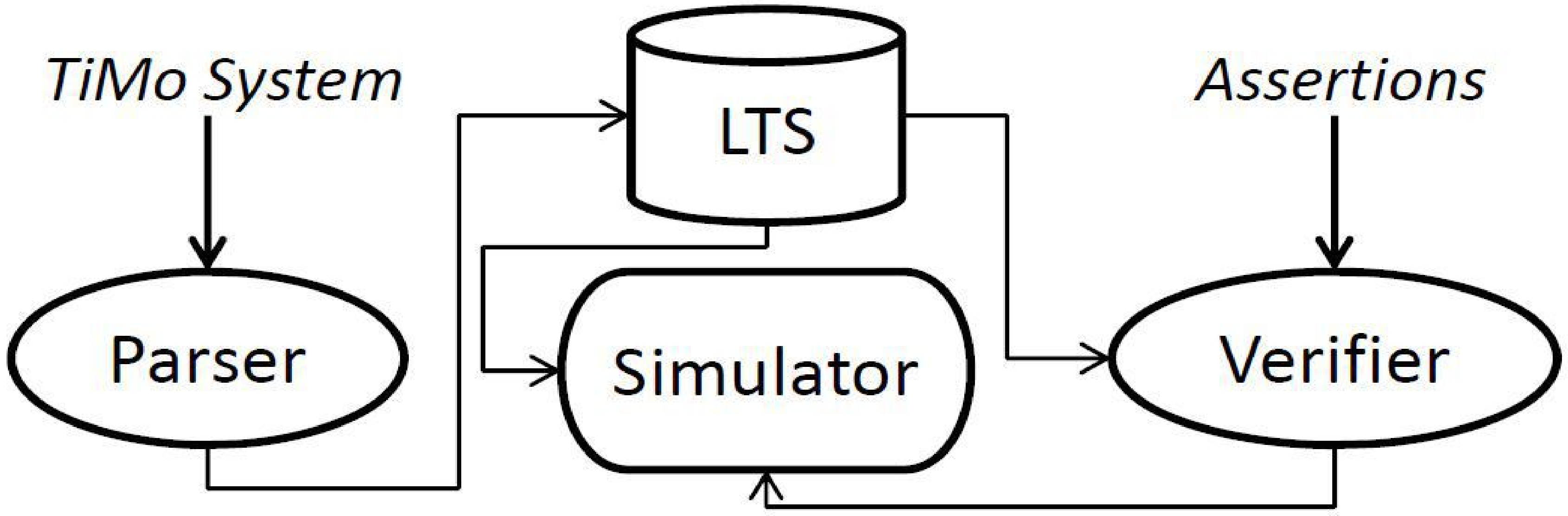

Distributed Algorithms, Web Services, Bio-systems, Security Protocols, Sensor Networks, etc.
Modeling
Concurrent Module
Real-time Module
Probabilistic Module
Web Service Module
...
Abstraction
Domain-specific Abstraction: data abstraction, zone abstraction, environment abstraction, etc.
Apply Operational Semantics
Symbolic Encoding
Intermediate
Explicit Representation:
Labeled Transition System
Timed Transition System
Markov Decision Process
Simulator
Symbolic Representation: Boolean Formulas
Binary Decision Diagram
CNF
Analysis
LTS Verification Algorithms
TTS Verification Algorithms
MDP Verification Algorithms
Counter-example Generator
Symbolic Verification Algorithms
SAT Solvers
Explicit Model Checking
Symbolic Model Checking

TiMo@PAT Framework

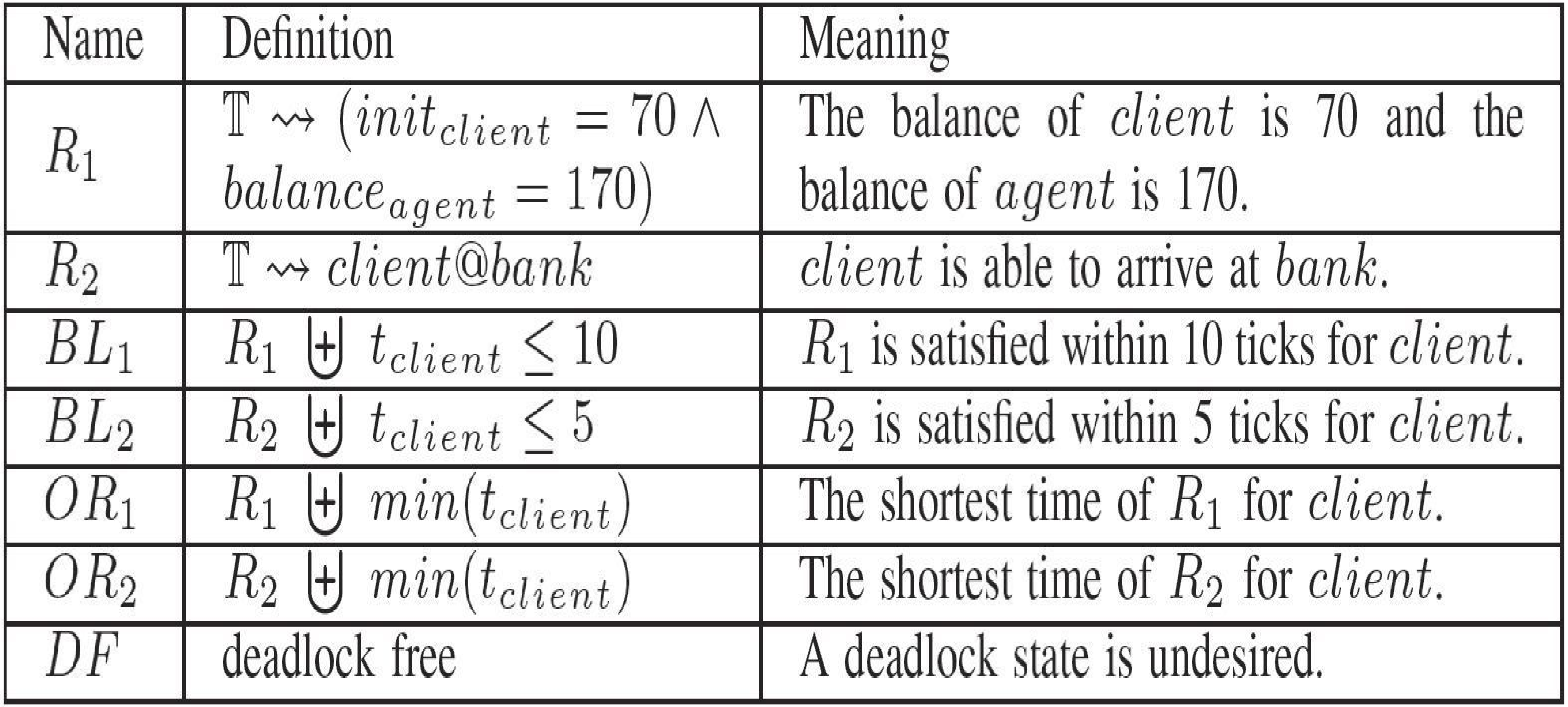
Modularized design

Over 2640 registered users, over 600 organizations

TiMo@PAT: automatic verification of TiMo systems

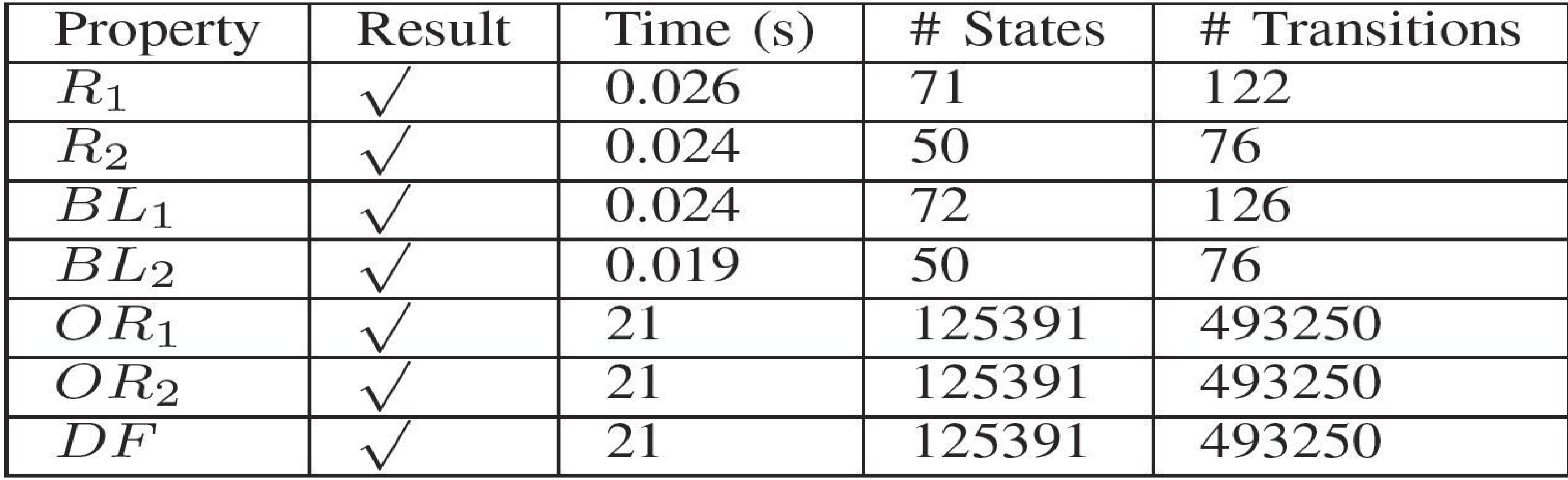


Properties Definitions



Name	Definition	Meaning
R_1	$\mathbb{T} \leadsto (init_{client} = 70 \wedge balance_{agent} = 170)$	The balance of *client* is 70 and the balance of *agent* is 170.
R_2	$\mathbb{T} \leadsto client@bank$	*client* is able to arrive at *bank*.
BL_1	$R_1 \uplus t_{client} \leq 10$	R_1 is satisfied within 10 ticks for *client*.
BL_2	$R_2 \uplus t_{client} \leq 5$	R_2 is satisfied within 5 ticks for *client*.
OR_1	$R_1 \uplus min(t_{client})$	The shortest time of R_1 for *client*.
OR_2	$R_2 \uplus min(t_{client})$	The shortest time of R_2 for *client*.
DF	deadlock free	A deadlock state is undesired.

Experiments Results



Property	Result	Time (s)	# States	# Transitions
R_1	$\surd$	0.026	71	122
R_2	$\surd$	0.024	50	76
BL_1	$\surd$	0.024	72	126
BL_2	$\surd$	0.019	50	76
OR_1	$\surd$	21	125391	493250
OR_2	$\surd$	21	125391	493250
DF	$\surd$	21	125391	493250

R1, R2: reachability properties;

BL1, BL2: bounded liveness properties;

OR1, OR2: optimized properties;

DF: deadlockfree

From TiMo to pTiMo (a quantitative approach)

- In order to allow a quantitative examination of behaviours, we add probabilities to TiMo, resulting in the new language pTiMo (probabilistic TiMo).

- Accordingly, pTiMo models are no longer labelled transitions systems (LTSs), like in TiMo, but instead they are labelled discrete-time Markov chains (DTMCs).

Adding quantitative aspects in pTiMo

Typical properties:

TiMo: "can a given system reach a certain state before *t1* time steps have elapsed at location *l1*?"

versus

pTiMo: "what is the probability that a given system reaches a certain state before *t1* time steps have elapsed at location *l1*?"

Sources of non-determinism

In pTiMo we treat the sources of non-determinism in the following way:

1. split complete computational steps into:
   - a part containing only potential movements
   - a part containing only potential communications
2. define discrete probability distributions for each part, individually
3. define discrete probability distributions for location selections
4. combine the resulting probability distributions into joint distributions

Probabilistic logic PLTM for pTiMo

As a means of investigating the behaviour of pTiMo networks, we define a new logic, named PLTM (Probabilistic Logic for Timed Mobility).

PLTM includes features such as:

- properties for short-run and long-run behaviour
- explicit references to locations and processes
- temporal constraints over local clocks, both finite and infinite
- complex action guards over multisets of transitions (i.e.,complete computational steps)

Examples of PLTM properties

Some properties which can be verified in PLTM:

- "with probability greater than 0.5, the process *P1* will communicate at location *l1*, on channels *a1* or *a3*, before 3 time steps have elapsed at location *l1*, and 4 time steps have elapsed at location *l2*“

- "the long-run probability that no movement occurs during a complete transition is less than 0.3"

PLTM Syntax

By employing atomic propositions, action guards and temporal constraints, the syntax of PLTM can be defined in terms of the following path and steady-state properties:

$$\phi_P ::= \texttt{true} \mid prop \mid \neg\phi_P \mid \phi_{P1}\wedge\phi_{P2} \mid [\phi_{P1}\mathbf{U}_{\phi_A}\phi_{P2}]^{\leq t}_{\geq p} \mid [\phi_{P1}\mathbf{U}_{\phi_A}\phi_{P2}]^{\leq t}_{> p}$$

$$\phi_S ::= [\mathbf{S}\phi_P]_{\geq p} \mid [\mathbf{S}\phi_P]_{> p} \mid [\mathbf{S}\phi_A]_{\geq p} \mid [\mathbf{S}\phi_A]_{> p}$$

where *prop* is an atomic proposition, *p* is a probability,

ϕ_A is a guard over complete transitions, *t* is a temporal constraint.

rTiMo: a real-time extension of TiMo

A real-time extension of TiMo named rTiMo, uses real-time and explicit timeouts

- it is useful for expressing certain temporal properties of multi-agent systems with migration and time constraints.

In rTiMo, the discrete transitions caused by performing actions with timeouts are alternated with continuous transitions.

rTiMo: a real-time extension of TiMo

Although the syntax of rTiMo is quite close to that of TiMo, its semantics is different in many aspects:

- the number of semantic rules (higher in rTiMo),
- number of clocks,
- time nature (continuous or discrete),
- systems evolution.

(DMove) $$\frac{t \geq t' \geq 0}{l[[go^{\Delta t} l' \text{ then } P]] \overset{t'}{\rightsquigarrow} l[[go^{\Delta t - t'} l' \text{ then } P]]}$$

rTiMo vs TiMo

- ***deadline*** in rTiMo is a positive *real* number,
  while in TiMo it is a *natural* number;
- ***clock*** in rTiMo is a single global clock,
  while in TiMo there are local clocks (for each location);
- ***time step*** in rTiMo can have any length,
  while in TiMo it has length 1 (at each location);
- ***passage of time*** in rTiMo is performed by delay rules, in contrast with TiMo where in each location l there is a local function φl that is used to decrement all timers by 1 at location l;

rTiMo vs TiMo

- ***evolution step*** in rTiMo is a sequence of individual actions followed by the passing of time, in contrast with TiMo where an evolution step is a sequence of individual actions happening at the same location l, followed by the passing of time and elimination of all special symbols $\circledS$ at location l;
- $\circledS$ is a purely technical notation used in the formalisation of the structural operational semantics of TiMo; intuitively $\circledS$ P specifies a process P which is temporarily stalled, and so cannot execute any action.

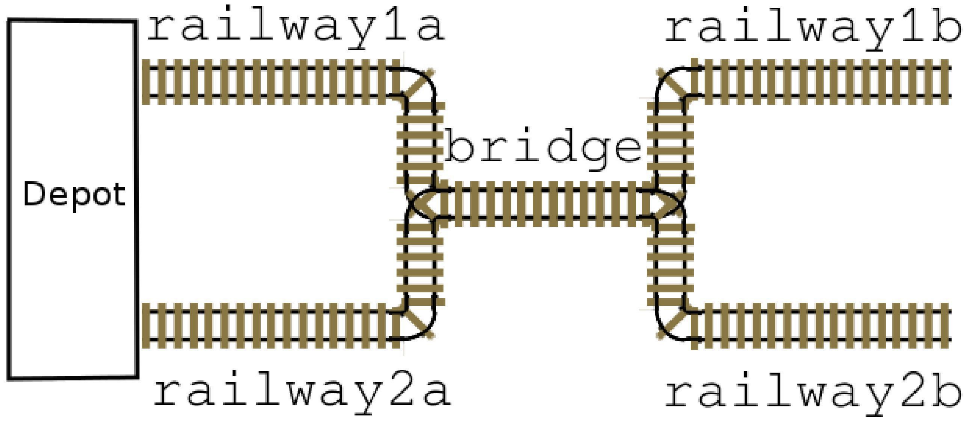
Example

The use of rTiMo for specifying critical systems is illustrated by considering a railway bridge controller, a real-time problem concerned with the control of accessing a mobile bridge by several trains leaving a depot according to the rule that the bridge can be accessed only by one train at a time.

Example

We use a small toy example in which the system is defined as:

- a number of three trains,
- two railways
- a bridge that can be up or down



The initial system is described in rTiMo by:

railway1a[[train1 | train3]] | railway1b[[0]]
| railway2a[[train2]] | railway2b[[0]]
| bridge[[operate | control1]]

rTiMo: a real-time extension of TiMo

We established a relationship between rTiMo and timed automata

- allowing the use of model checking capabilities provided by UPPAAL to verify several temporal properties of distributed networks with migrating and communicating processes described in rTiMo.

TiMo and Event-B

- We use the Event-B modelling method as the target for translating TIMO specifications

- We utilise the supporting Rodin platform of Event-B to verify system properties using the embedded theorem-provers and model checkers.

- The main feature of our encoding include a generic model capturing the syntax and semantics of TIMO, together with a concrete model corresponding to each specific TIMO specification.

TiMo and Event-B

- State-transition mode (like ASM, B, Z)
  - set theory as mathematical language
  - refinement as basic modeling approach
- Contexts
  - carrier sets (domains)
  - constants
  - axioms
- Machines
  - global variables
  - invariants
  - events that update the variables
- Events
  - local parameters
  - guards
  - actions

context C

sets s

constants c

axioms $A(s, c)$

machine M

variables v

invariants $I(v)$

events

init

$\mathsf{e}_1 \quad \cdots \quad \mathsf{e}_N$

$$\mathsf{e} \mathrel{\widehat{=}} \textsf{any } u \textsf{ where } G(u, v) \textsf{ then } v := E(u, v) \textsf{ end}$$

TiMo and Event-B

Once we have our Event-B model we can use the verification tools of Rodin:

- Model checking (ProB) can verify properties such as:
  - “the two customers cannot be at the same shop at the same time”, or
  - “once the customer left home, s/he will not go home”
  - (counterexamples were found for both of the above)

- Theorem proving can reason:
  - on the generic model - e.g. well-formedness (using invariants), or
  - on the concrete model

- Parameterised verification
  - using theorem proving, we can verify parameterised versions of the models (i.e. timers not fixed, but depend on the number of processes)
  - this is not feasible with model checking,
  - but not easy with theorem proving either (less than 40% of the proofs were discharged automatically)

Concluding Remarks

TiMo models distributed systems with time and mobility

- It has a simple syntax, but can model complex systems with respect to time and mobility.
- Timing constraints for migration and communication.
- Local clocks and maximal parallelism of actions.
- Security aspects expressed by dynamic access permissions (access permissions may be lost or gained by processes when moving).
- An operational semantics and formal results.
- A sound and complete system for safe communication and migration in open networks.

Concluding remarks

- pTiMo allows probabilistic behaviour of TiMo networks by solving the non-determinism involved in movement, communication and selection of active locations.
- PLTM: a probabilistic temporal logic for pTiMo
  - check properties with explicit reference to locations and processes
  - impose temporal constraints over local clocks (i.e., finite or infinite upper bounds, for each location independently)
  - define complex action guards over multisets of actions found in other logics
- A link between rTiMo and timed automata allows model checking by UPPAAL to verify temporal properties of distributed networks with migration and communication.

Many Thanks to Collaborators!

- G. Ciobanu, **M. Koutny**. Modelling and Verification of Timed Interaction and Migration. In Fundamental Approaches to Software Engineering, Lecture Notes in Computer Science, vol.4961, 215–229, 2008.

- G. Ciobanu, **M. Koutny**. Timed Migration and Interaction with Access Permissions. In 17th International Symposium on Formal Methods, Lecture Notes in Computer Science, vol.6664, 293–307, 2011.

Maciej Koutny (Newcastle University)

- G. Ciobanu, **M. Koutny**. T*imed Mobility in Process Algebra and Petri nets*. Journal of Logic and Algebraic Programming, vol.80(7), 377–391, 2011.

- B. Aman, G. Ciobanu, **M. Koutny**. *Behavioural Equivalences over Migrating Processes with Timers.* FMOODS/FORTE 2012, Lecture Notes in Computer Science, vol.7273, 52–66, 2012.

- G. Ciobanu, **M. Koutny**. PerTiMo: A Model of Spatial Migration with Safe Access Permissions. Comput. Journal vol. 58(5), 1041-1060, 2015.

Bogdan Aman (IIT, Iasi)

- **B. Aman**, G. Ciobanu, M. Koutny. *Behavioural Equivalences over Migrating Processes with Timers.* FMOODS/FORTE 2012, Lecture Notes in Computer Science, vol.7273, 52–66, 2012.

- **B. Aman**, G. Ciobanu. *Real-Time Migration Properties of rTiMo Verified in Uppaal*. SEFM 2013, Lecture Notes in Computer Science, vol.8137, 31-45, 2013.

- **B. Aman**, G. Ciobanu. *Timed Mobility and Timed Communication for Critical Systems*. FMICS 2015, Lecture Notes in Computer Science, vol. 9128, 146-161, 2015.

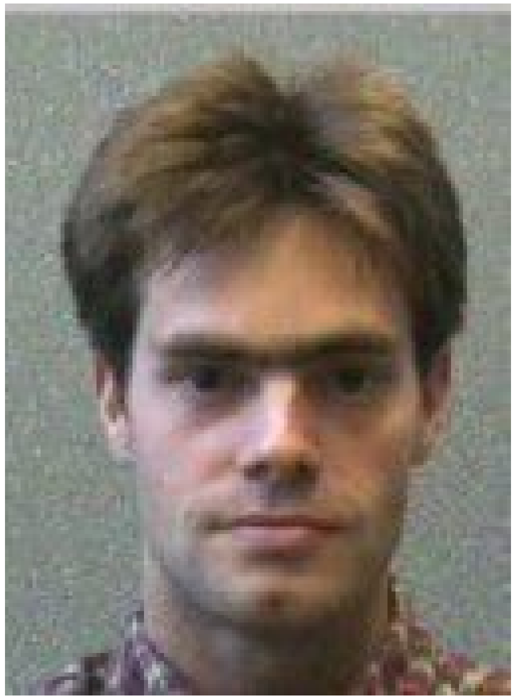
Calin Juravle (Google, London)

- G. Ciobanu, **C. Juravle**. A Software Platform for Timed Mobility and Timed Interaction. FORTE / FMOODS, Lecture Notes in Computer Science, vol.5522, 106–121, 2009.
- G. Ciobanu, **C. Juravle**. Mobile Agents with Timers, and Their Implementation. Intelligent Distributed Computing IV. Studies in Computational Intelligence, vol.315, Springer, 229–239, 2010.
- G. Ciobanu, **C. Juravle**. Flexible Software Architecture and Language for Mobile Agents. *Concurrency and Computation: Practice and Experience*, vol.24(6), 559–571, 2012.

Jason Steggles (Newcastle University)

- G. Ciobanu, M. Koutny, **J. Steggles**. *A Timed Mobility Semantics Based on Rewriting Strategies.* SEFM 2012, Lecture Notes in Computer Science, vol.7504, 141–155, 2012.

- G. Ciobanu, M. Koutny, **J. Steggles**. *Strategy based semantics for mobility with time and access permissions*. Formal Aspects of Computing vol. 27(3): 525-549, 2015.



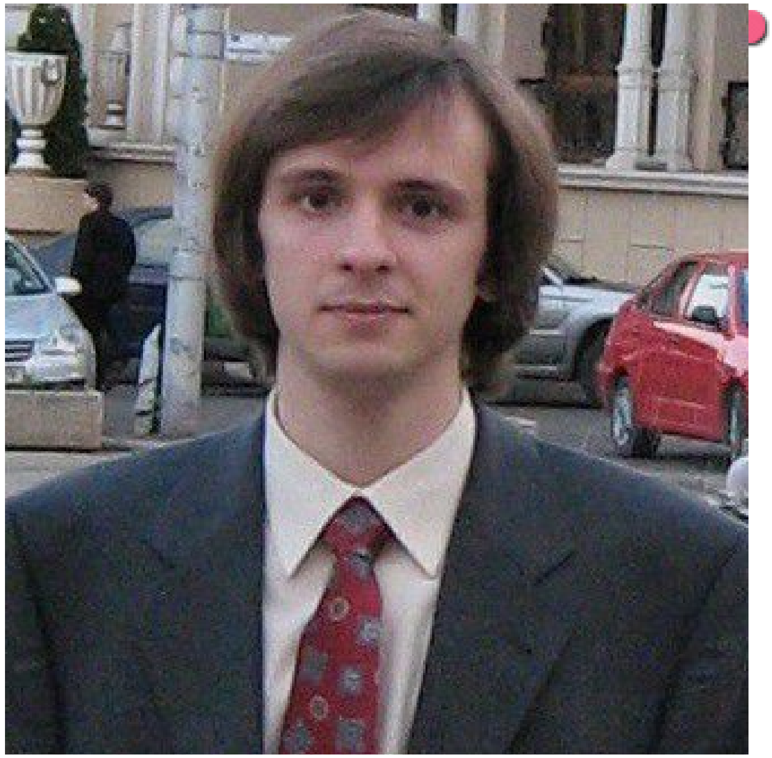
Thai Son Hoang (Hitachi Ltd., Japan)

- G. Ciobanu, **T.S. Hoang**, A. Stefanescu. *From TiMo to Event-B: Event-Driven Timed Mobility.* ICECCS 2014 (best paper award).

Alin Stefanescu (Bucharest, Romania)

- G. Ciobanu, T.S. Hoang, **A. Stefanescu**. *From TiMo to Event-B: Event-Driven Timed Mobility*. ICECCS 2014 (best paper award).

Armand Rotaru (UCL, London)



- G. Ciobanu, **A. Rotaru**. A Probabilistic Logic for PTIMO . In 10th International Colloquium on Theoretical Aspects of Computing, Lecture Notes in Computer Science, vol.8049, 141–158, 2013.

Manchun Zheng (NUS, Singapore)

- G. Ciobanu, **M. Zheng**. Automatic Analysis of TIMO Systems in PAT. In 18th International Conference on Engineering of Complex Computer Systems, 121-124, IEEE, 2013.

Thank You!