Model-Based Development and Validation of Multirobot Cooperative System

Jüri Vain
Dept. of Computer Science
Tallinn University of Technology

Syllabus

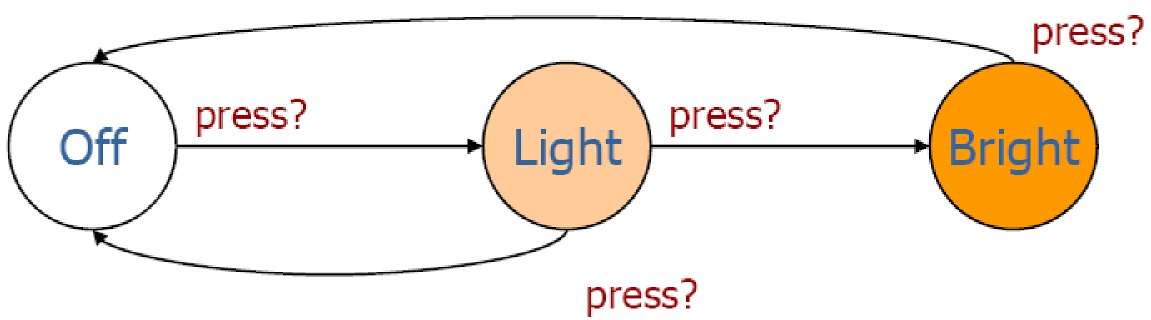
- **Monday morning: (9:00 – 12.30)**
  - 9:00 – 9:45 Introduction
  - 10:00 – 11:30 Hands-on exercises I: Uppaal model construction
  - 11:45 – 12:30 Theoretical background I: XTA semantics,
- **Lunch** 12.30 – 13.30
- **Monday afternoon: (13:30 – 16:30)**
  - 13.30 – 14:15 Applications I: model learning for Human Addaptive Scrub Nurse Robot
  - 14.30 – 15.15 Theoretical background II: model checking
  - 15.30 – 16:15 Hands-on exercises II: model checking
- **Tuesday morning: (9:00 – 12.30)**
  - 9:00 – 9:45 Theoretical background III: Model based testing
  - 10:00 – 10:45 Applications II: reactive planning tester
  - 11:00 – 12:30 Hands-on exercises III (model refinement)

Lecture #L2 : Model construction
Lecture Plan

- Extended Timed Automata (XTA) (slides by Brien Nielsen, Aalborg Univ.)
  - Syntax
  - Semantics (informally)
  - Example
- Learning XTA
  - Motivation: why learning?
  - Basic concepts
  - Simple Learning Algorithm
  - Adequacy of learning: Trace equivalence

Timed automata

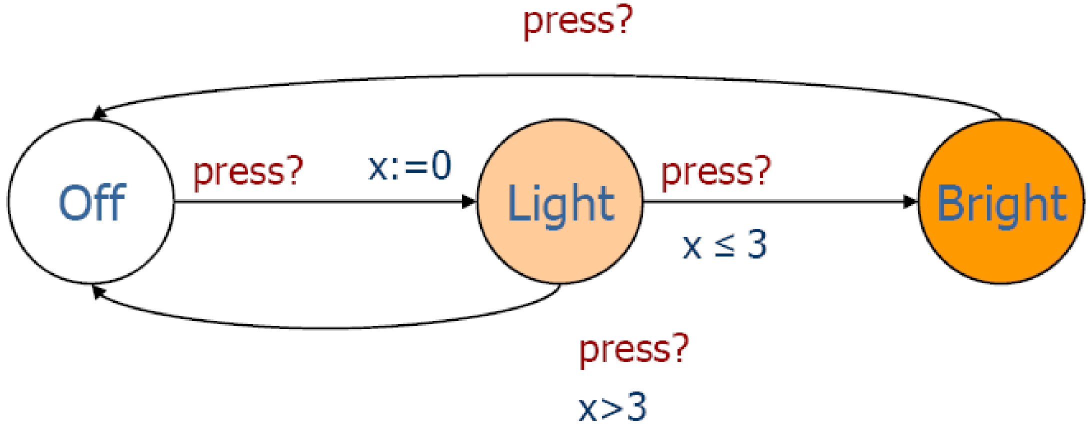
Dumb Light Control



WANT: if press is issued twice quickly then the light will get brighter; otherwise the light is turned off.

Timed automata

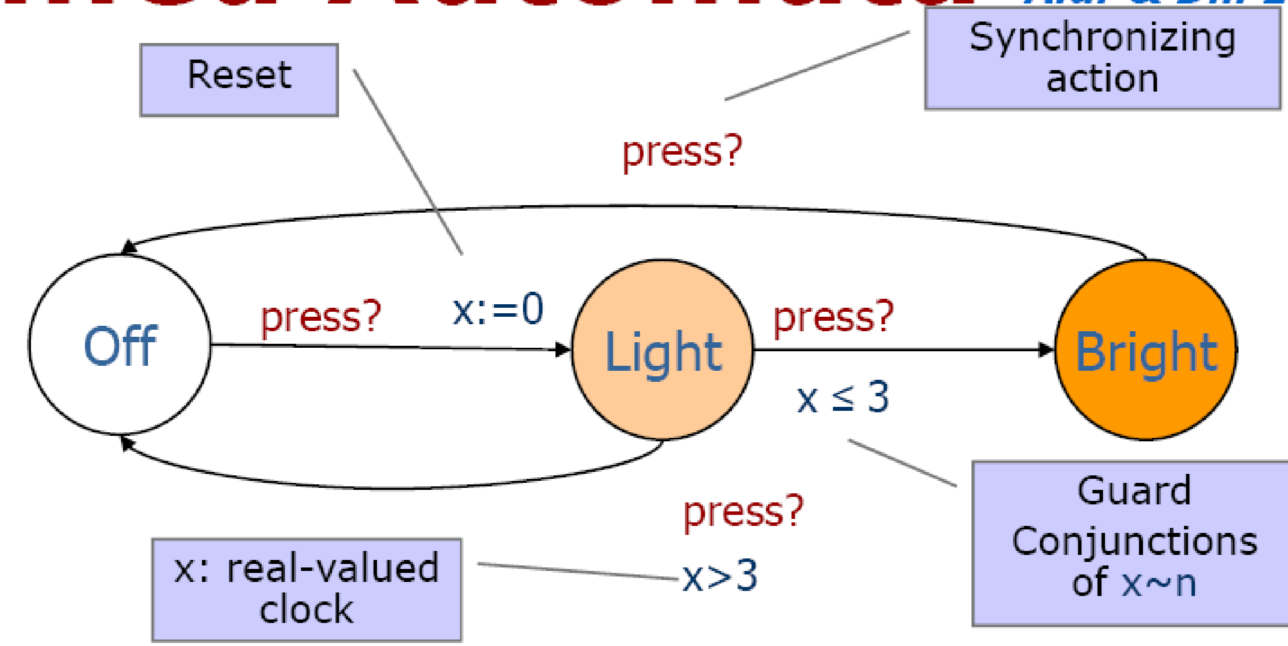
Dumb Light Control *Alur & Dill 1990*



Solution: Add real-valued clock **x**

Timed Automata

Alur & Dill 1990



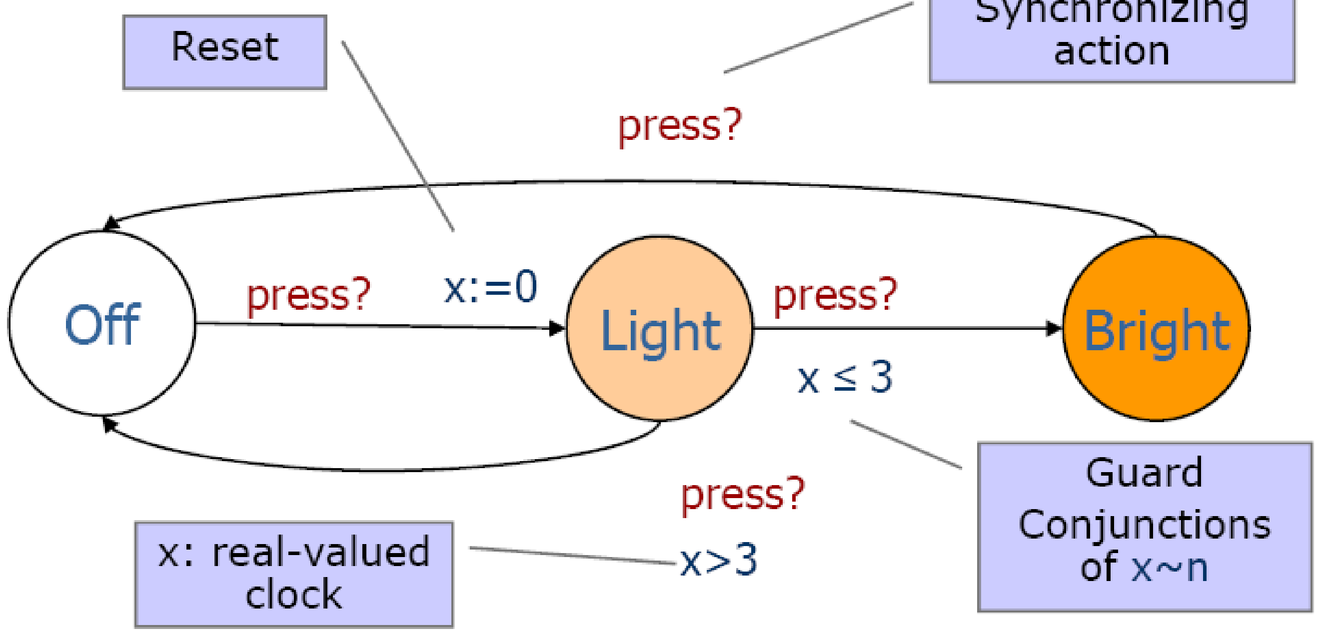
States:

(location , x=v) where v ∈ **R**

Transitions:

(Off , x=0)

Timed Automata *Alur & Dill 1990*



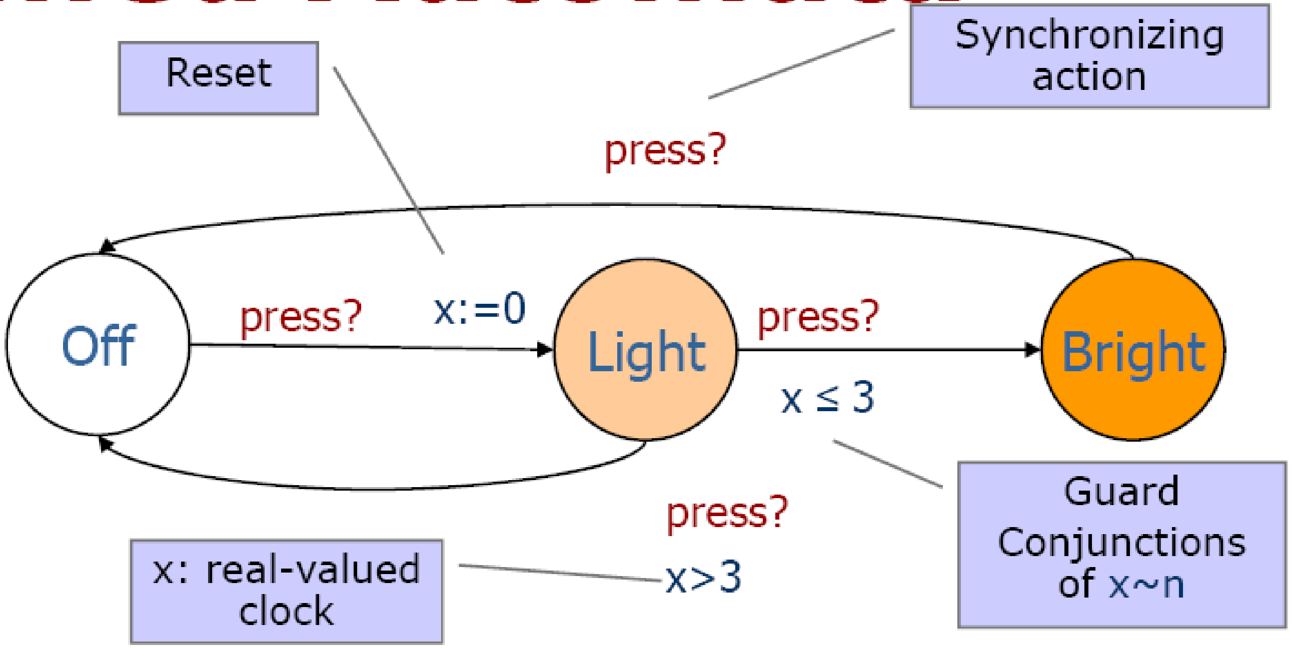
States:

(location , x=v) where v ∈ **R**

Transitions:

(Off , x=0)
delay 4.32 → (Off , x=4.32)

Timed Automata *Alur & Dill 1990*



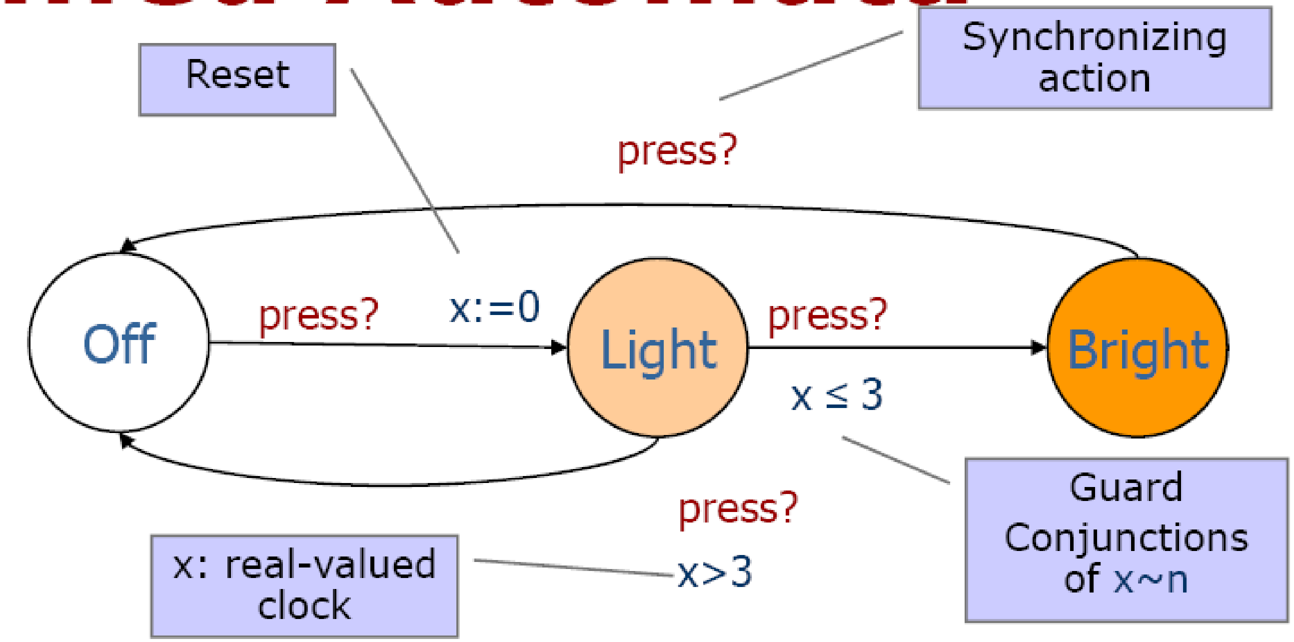
States:

(location , x=v) where v ∈ **R**

Transitions:

	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)

Timed Automata *Alur & Dill 1990*



States:

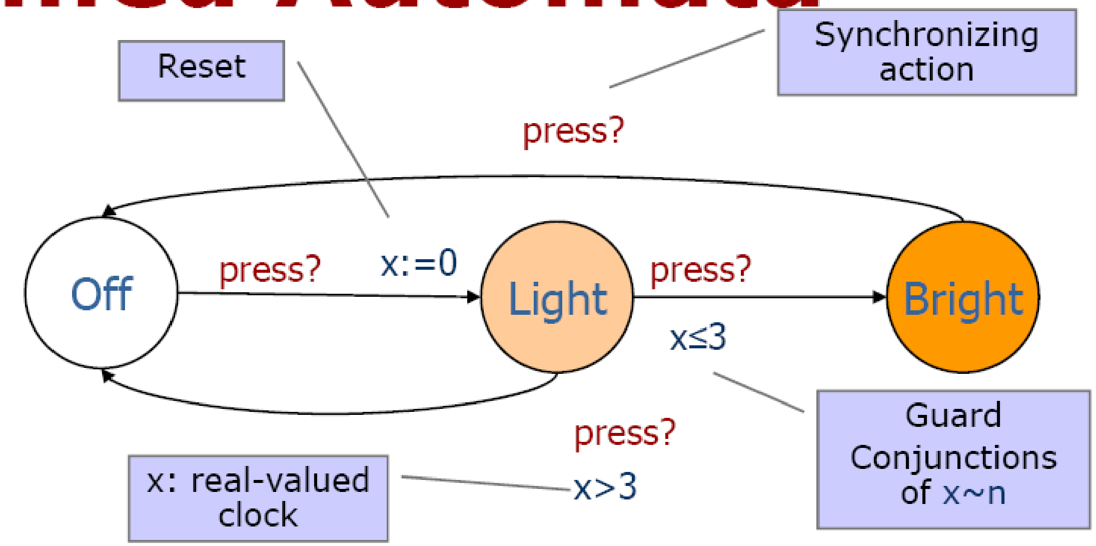
(location , x=v) where v ∈ **R**

Transitions:

	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 2.51	→ (Light , x=2.51)

Timed automata: Semantics

Timed Automata *Alur & Dill 1990*



States:

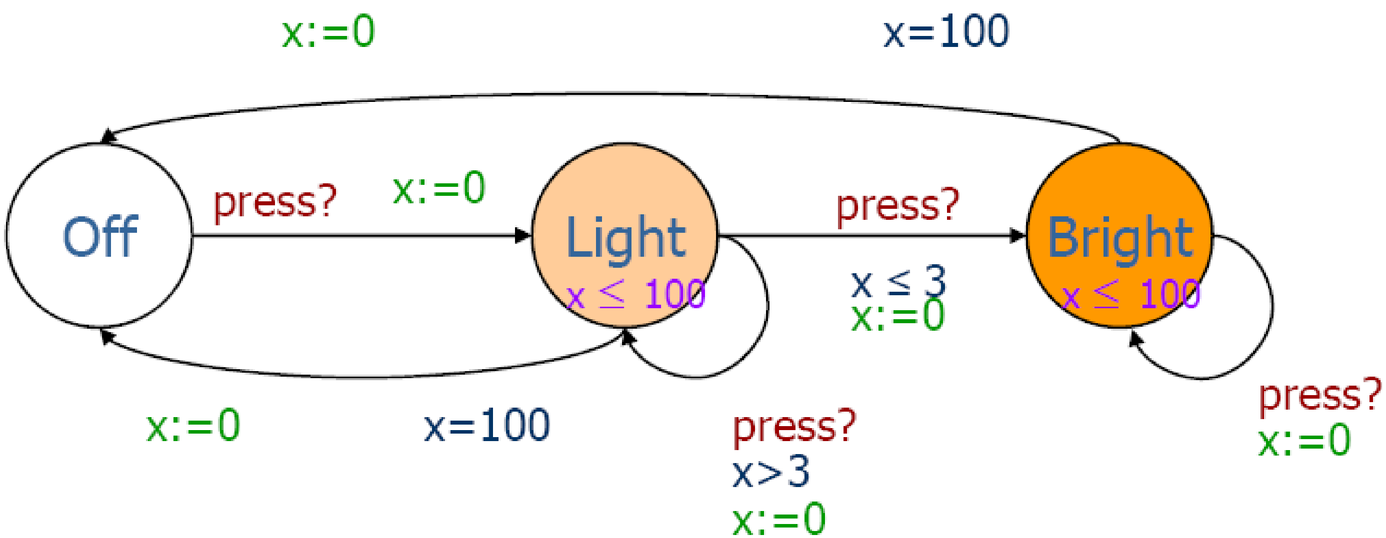
(location , x=v) where v ∈ **R**

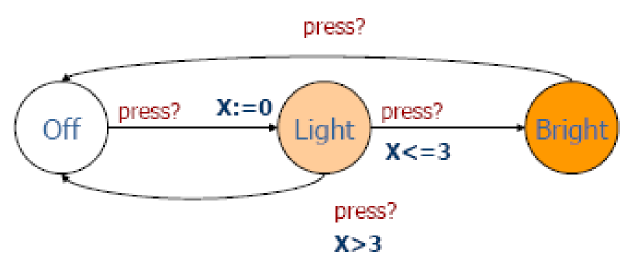
Transitions:

	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 2.51	→ (Light , x=2.51)
press?	→ (Bright , x=2.51)

Intelligent Light Control

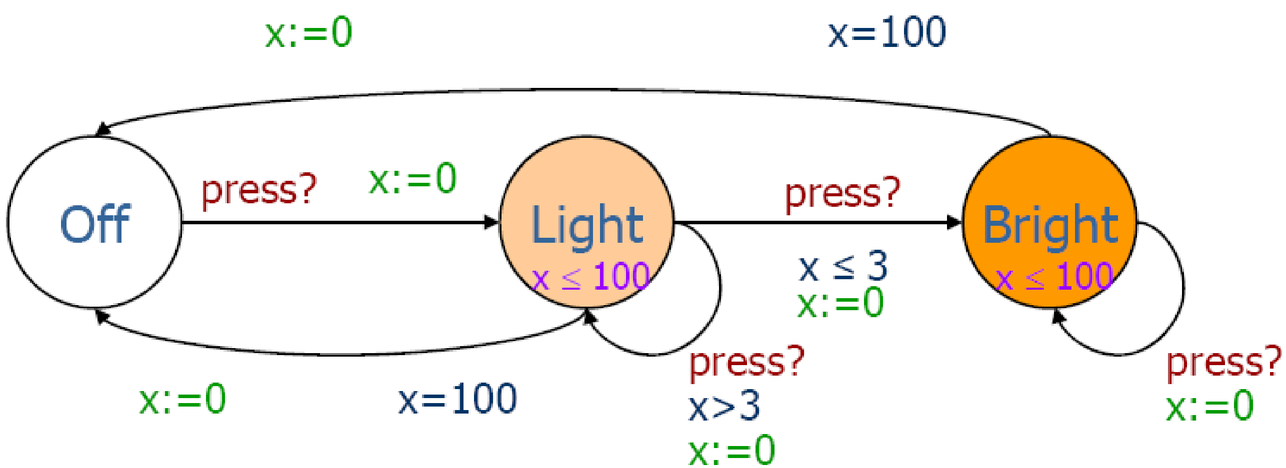
Using Invariants





Intelligent Light Control

Using **Invariants**



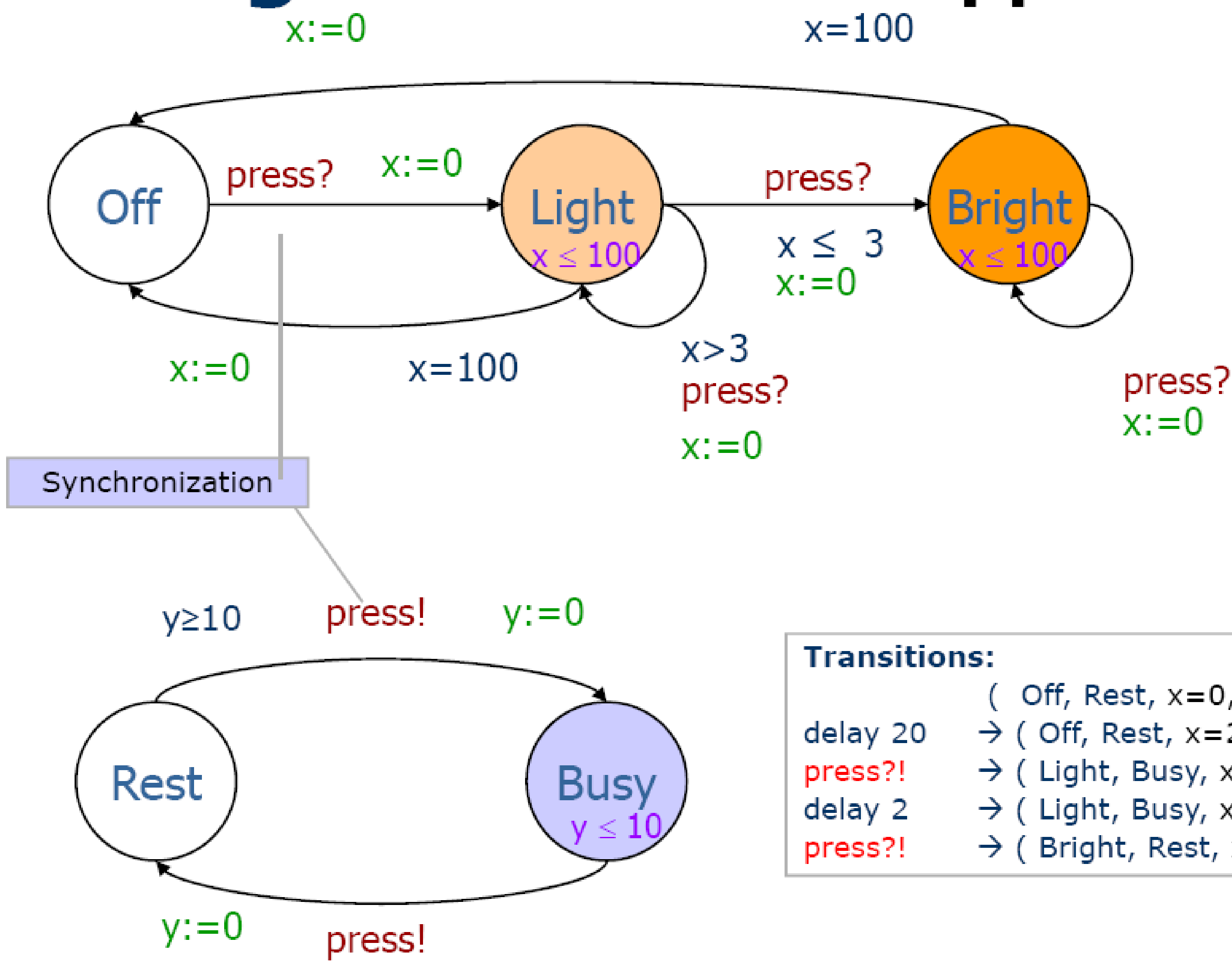
Transitions:

	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 4.51	→ (Light , x=4.51)
press?	→ (Light , x=0)
delay 100	→ (Light , x=100)
τ	→ (Off , x=0)

Note:
(Light , x=0) delay 103 → X

Invariants ensures progress

Light Controller || User



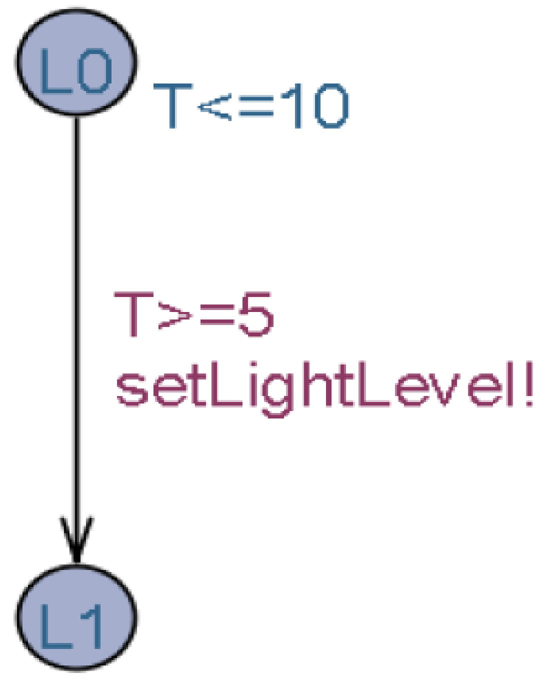
Transitions:

	(Off, Rest, x=0, y=0)
delay 20	→ (Off, Rest, x=20, y=20)
press?!	→ (Light, Busy, x=0, y=0)
delay 2	→ (Light, Busy, x=2, y=2)
press?!	→ (Bright, Rest, x=0, y=0)

Timing Uncertainty

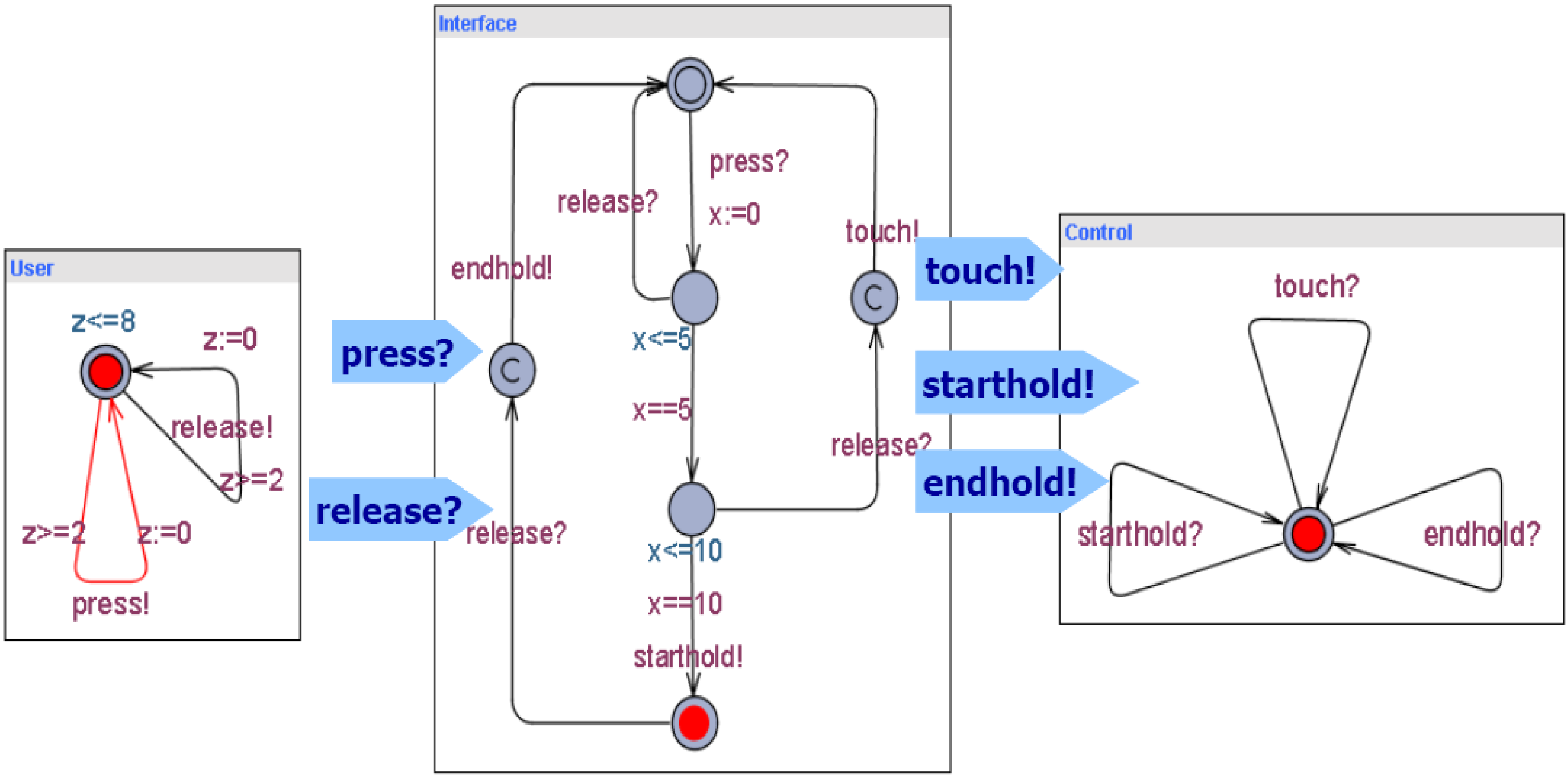
- Unpredictable or variable
  - response time,
  - computation time
  - transmission time etc:

- Initially T=0



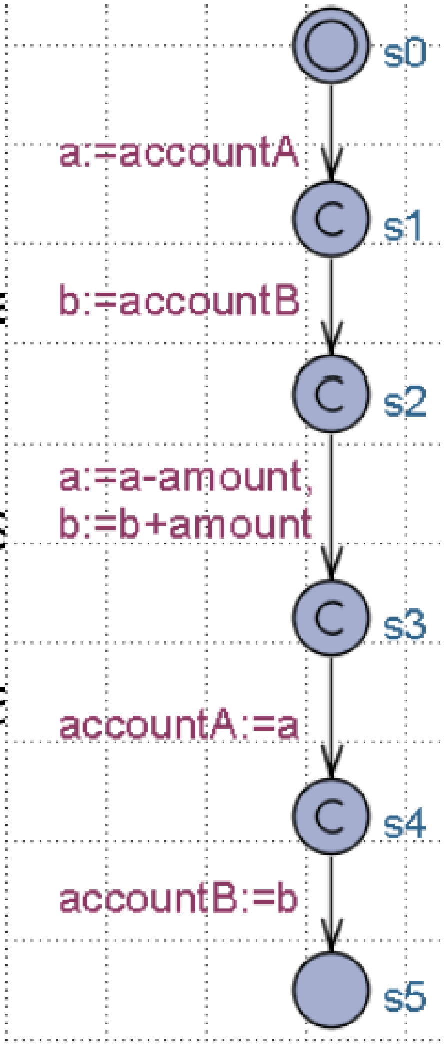
LightLevel must be adjusted between 5 and 10

Light Control Network



Comitted Locations

- Locations marked **C**
  - ***No delay*** in committed location.
  - Next transition must involve automata in ***committed location.***
- Handy to model atomic sequences
- The use of committed locations <u>reduces</u> the number of states in a model, <u>and</u> allows for more space and time efficient analysis.
- S0 to s5 executed atomically



Urgent Channels and Locations

- Locations marked **U**
  - ***No delay*** in committed location.
  - Interleaving permitted
- Channels declared "**`urgent chan`**"
  - Time doesn't elapse when a synchronization is possible on a pair of urgent channels
  - Interleaving allowed

Other Uppaal features

- Bounded domain
  - Int [1..4] a;
- C-like data-structures and user defined functions in declaration section
  - structs, arrays, and typedef
- **`select a:T`** construct
- `Forall`, `exists` in expr
- Scalar sets (for giving unique ID’s)
- Process and channel **priorities**
- Value passing (emulation)

See Uppaal Help for details!

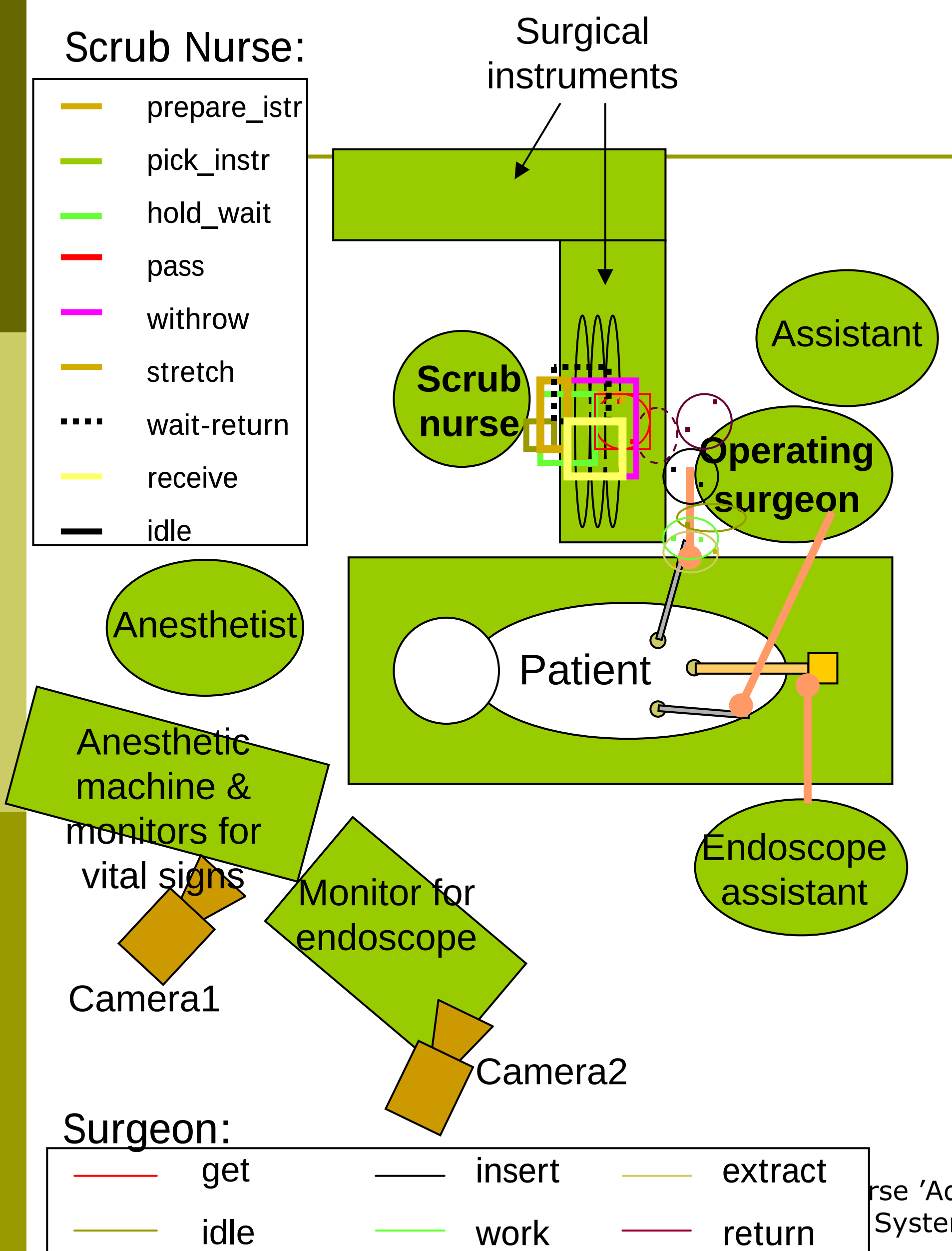
Learning XTA: about terminology

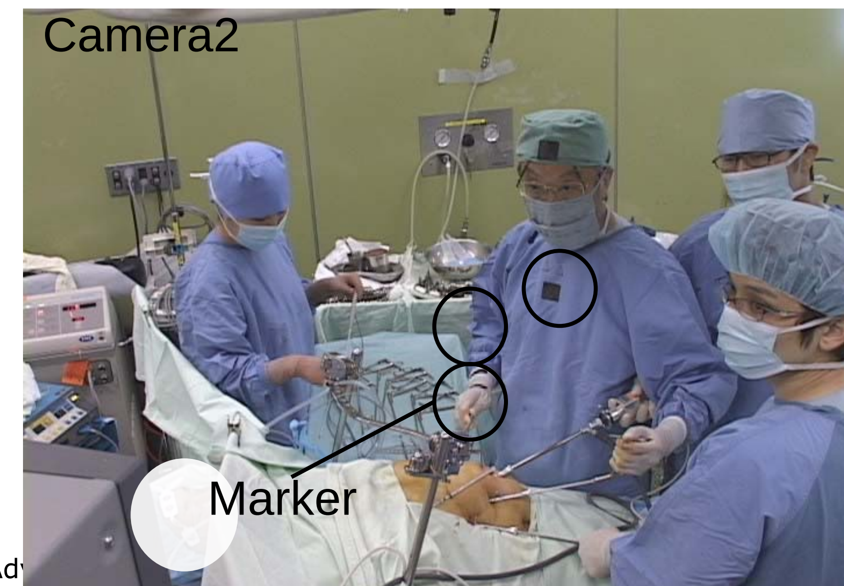
- General term is Machine Learning
  - Passive *vs* active
  - Supervised vs unsupervised
  - Reinforcement learning (reward guided)
  - Computational structures used:
    - FSA
    - Hidden Markov Model
    - Kohonen Map
    - NN
    - Timed Automata
    - etc

Learning XTA (lecture plan)

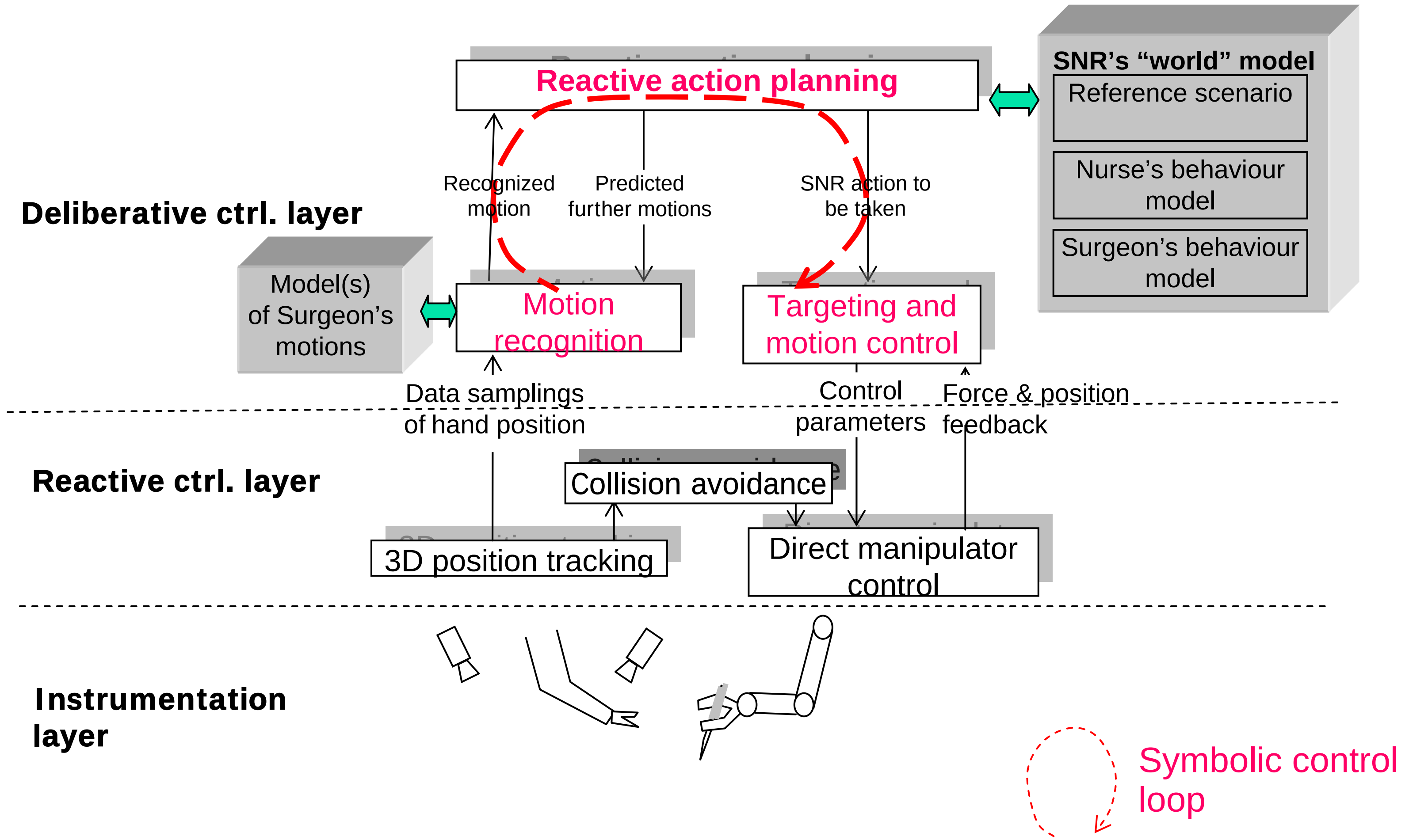
- Problem context
- Simplifying assumptions
  - I/O observability
  - Generally non-deterministic models
  - Fully observable (output determinism)
- The learning algorithm
- Estimating the quality of learning

Problem context: SNR scene and motion analysis

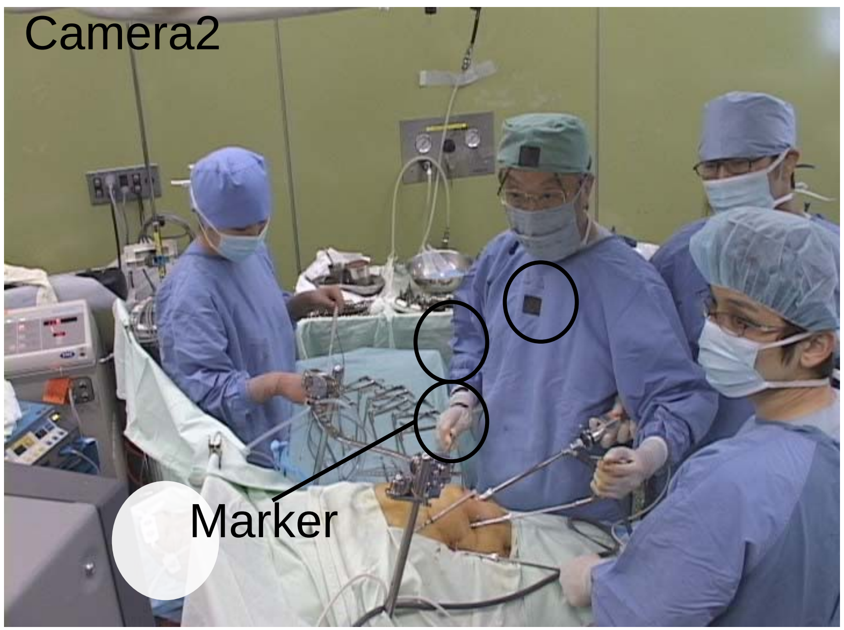




SNR Control Architecture

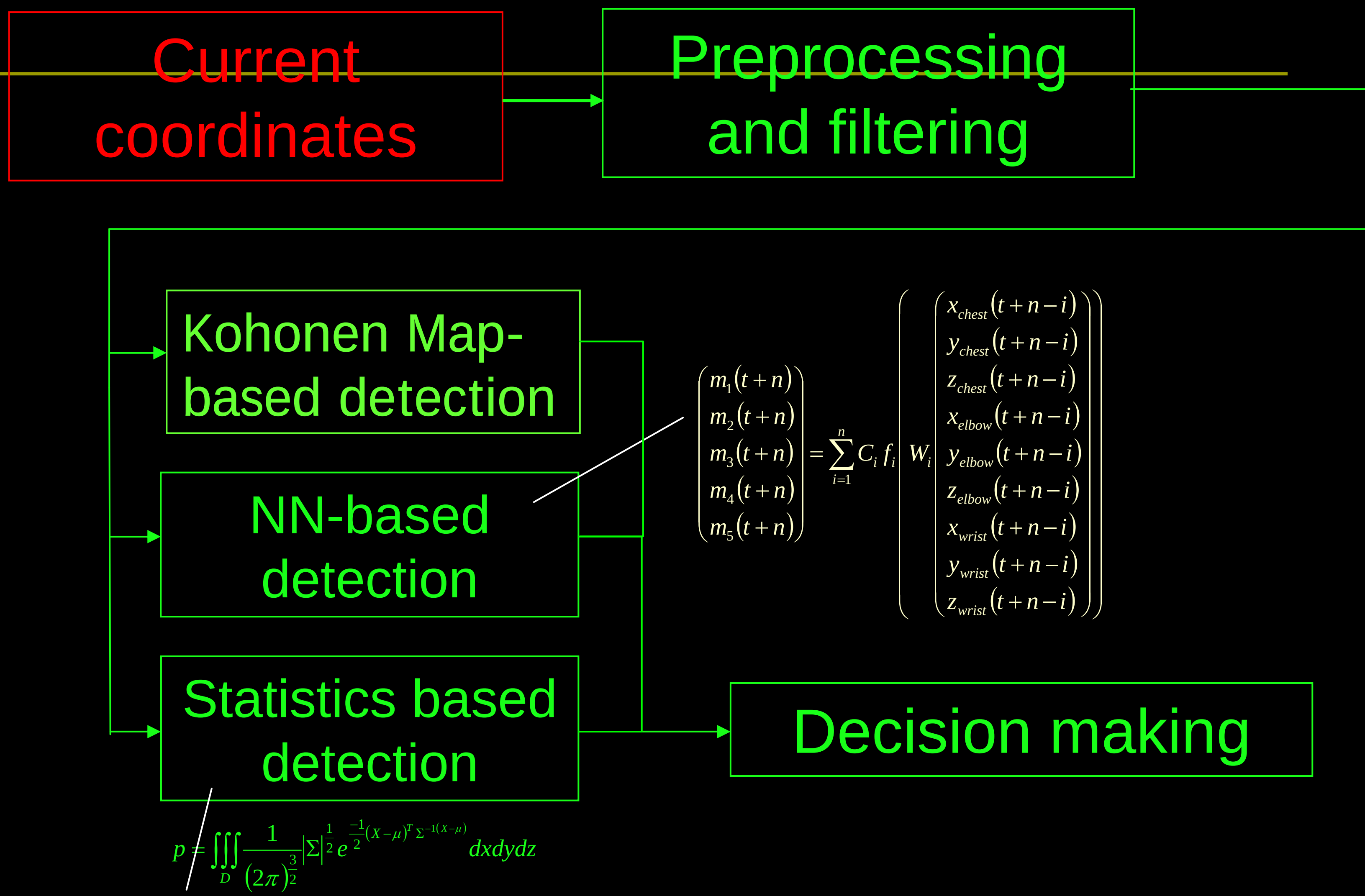


Scrub Nurse Robot (SNR): Motion analysis

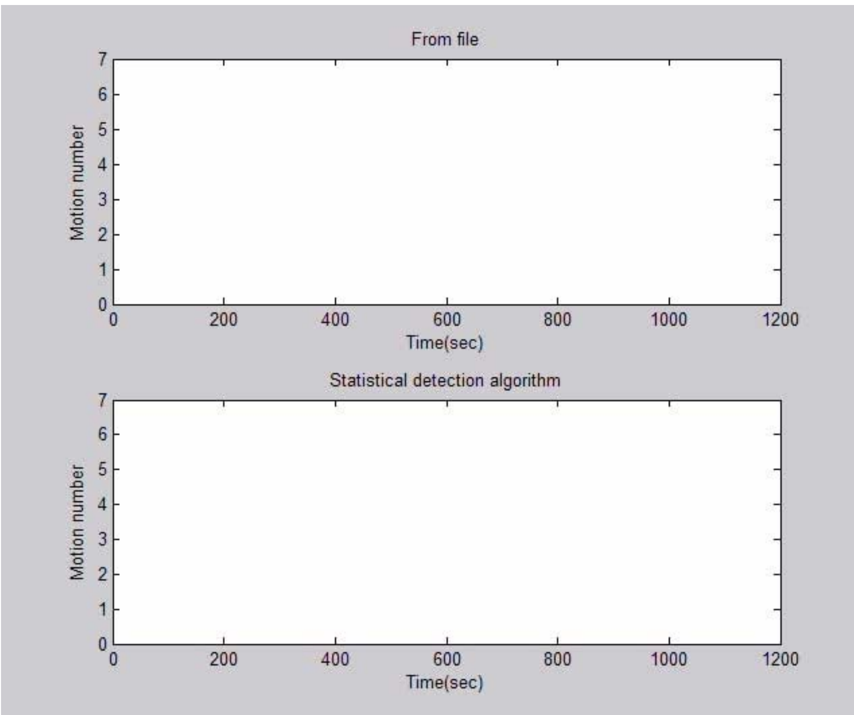


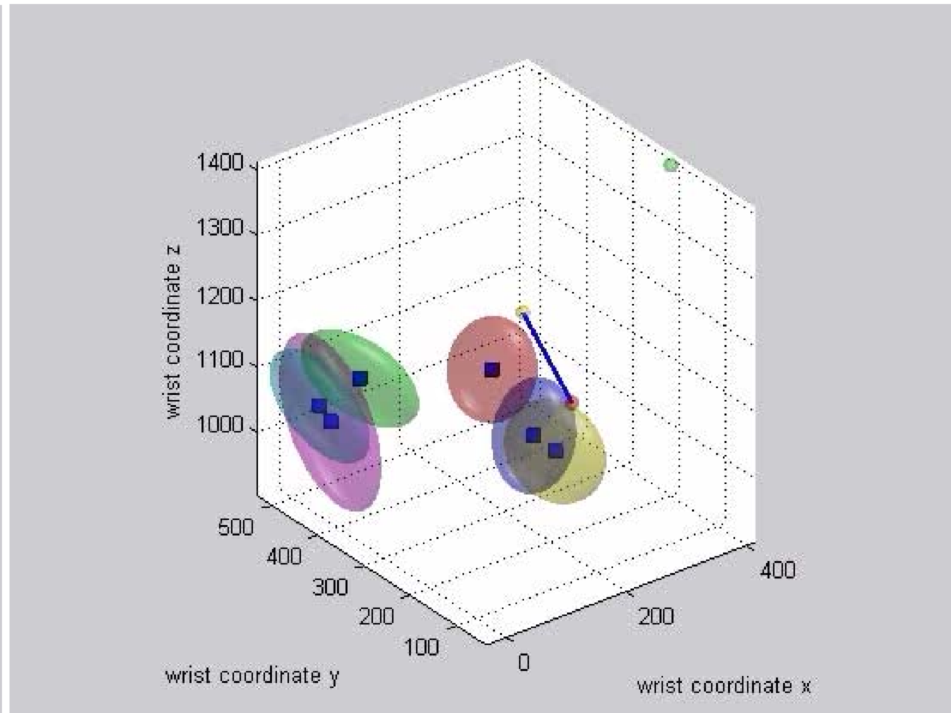
Photos from CEO on HAM, Tokyo Denki University

Motion recognition



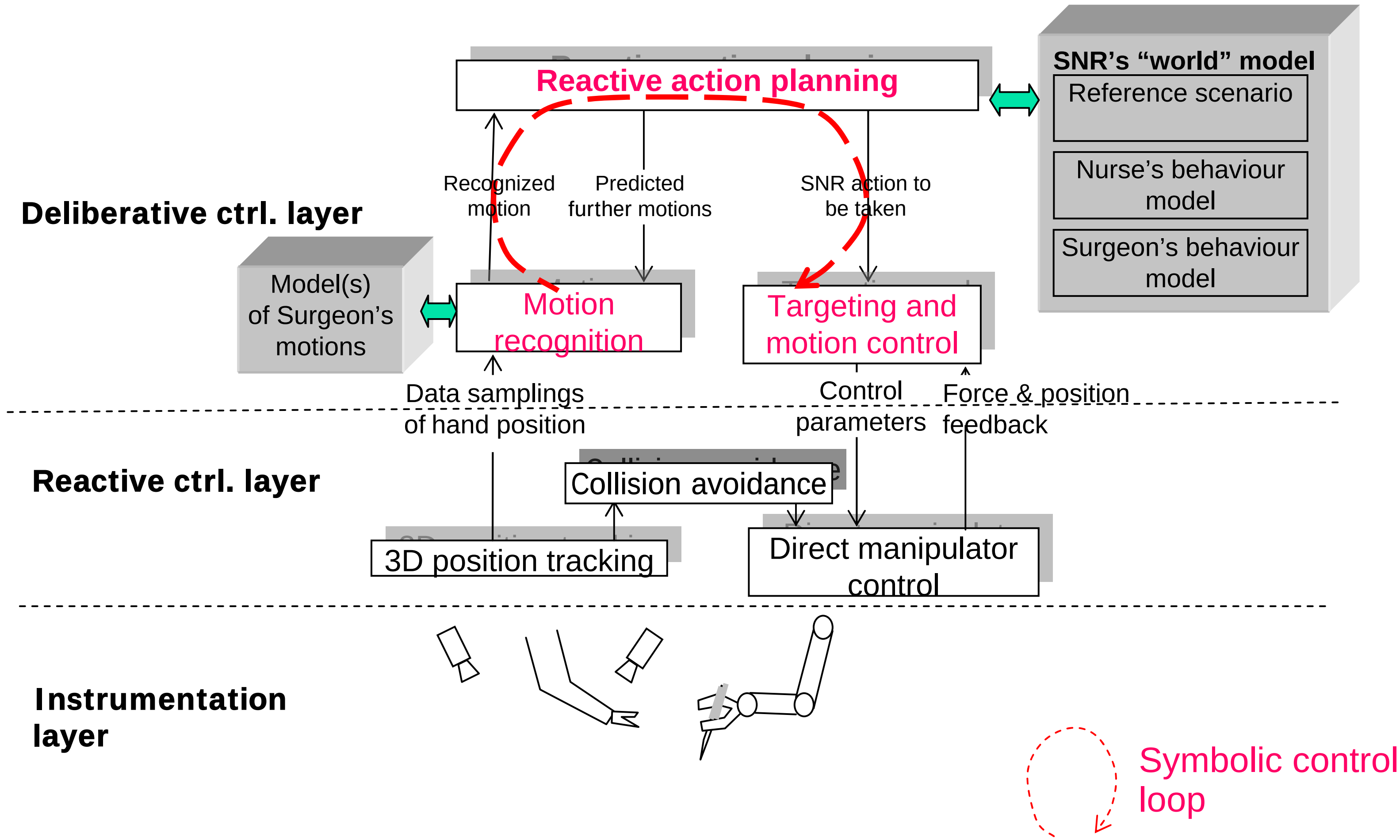
Motion recognition using statistical models





| - working->extracting, | - extracting->passing, | - passing->waiting, | - waiting->receiving, | - receiving->inserting, | - inserting->working

SNR Control Architecture



(Timed) automata learning algorithm

- Input:
  - Time-stamped sequence of observed i/o events (timed trace *TTr(Obs)*)
  - Observable inputs/outputs X_{Obs} of actors
  - Rescaling operator $\mathcal{R}$: $X_{Obs} \longrightarrow X_{XTA}$
    - where X_{XTA} is a model state space
  - Equivalnece relation "~" defining the quotient state space $X /_{\sim}$
- Output:
  - Extended (Uppaal version) timed automaton *XTA* s.t.

$TTr(XTA) = \mathcal{R}(TTr(Obs)) /_{\sim}$ % = equivalence of traces

Algorithm 1: model compilation (one learning session)

- *Initialization*

$L \leftarrow \{l_0\}$	% L –	set of locations, l_0 – (auxiliary) initial location
$T \leftarrow \varnothing$	% T –	set of transitions
$k,k' \leftarrow 0,0$	% k,k' –	indexes distinguishing transitions between same location pairs
$h \leftarrow l_0$	% h –	history variable storing the id of the previous motion
$h' \leftarrow l_0$	% h' –	variable storing the id of the motion before previous
$h_{cl} \leftarrow 0$	% h_{cl} –	clock reset history
$l \leftarrow l_0$	% l –	destination location of the current switching event
$cl \leftarrow 0$	% cl –	clock variable of the automaton being learned
$g\_cl \leftarrow \varnothing$	% g_cl -	3D matrix of clock reset intervals
$g\_x \leftarrow \varnothing$	% g_x -	4D matrix of state intervals that define switching cond.s

Encode new motion:

- Create a new eq.class (lines 5–11)

Encode motion previously observed:

- Match with existing eq.class
- Extend existing eq.class
- Create a new eq.class

```
1: while E ≠ ∅ do
2:     e ← get(E )                          % get the motion switching event record from buffer E
3:     h'← h, h ← l
4:     l ← e[1], cl ← (e[2] - h_cl), X ← e[3]
5:     if l ∉ L then                        % if the motion has never occurred before
6:         L ← L ∪ {l},
7:         T ← T ∪ {t(h,l,1)}               % add transition to that motion
8:         g_cl(h,l,1) ← [cl, cl]           % add clock reset point in time
9:         for all x_i ∈ X do
10:          g_x(h,l,1,x_i) ← [x_i, x_i]    % add state switching point
11:        end for
12:    else                                 % if switching e in existing equivalence class
13:        if ∃k∈ [1,|t(h,l,.)|], ∀ x_i∈ X,: x_i ∈ g_x(h,l,k,x_i) ∧ cl ∈ g_cl(h,l,k) then
14:         goto 34
15:        else                             % if switching e extends the equival. class
16:            if ∃k∈ [1,|t(h,l,.)|], ∀ x_i∈ X:  x_i ∈ g_x(h,l,k,x_i)^⇳Ri ∧ cl ∈ g_cl(h,l,k)^⇳Rcl
17:            then
18:                if cl < g_cl(h,l,k)^-  then g_cl(h,l,k) ← [cl, g_cl(h,l,k)^+] end if
19:                if cl > g_cl(h,l,k)^+ then g_cl(h,l,k) ← [g_cl(h,l,k)^-, cl]  end if
20:                for all x_i ∈ X  do
21:                        if x_i < g_x(h,l,k,x_i)^-   then g_x(h,l,k,x_i) ← [x_i, g_x(h,l,k,x_i)^+] end if
22:                        if x_i > g_x(h,l,k,x_i)^+  then g_x(h,l,k,x_i) ← [g_x(h,l,k,x_i)^-, x_i] end if
23:          end for
24:          else              % if switching e exceeds allowed limits of existing eqv. class
25:          k ←|t(h,l,.)| +1
26:          T ← T ∪ {t(h,l,k)}             % add new transition
27:          g_cl(h,l,k) ← [cl, cl]         % add clock reset point in time
28:          for all x_i ∈ X do
29:              g_x(h,l,k,x_i) ← [x_i, x_i]  % add state switching point
30:          end for
31:        end if
32:    end if
33:    a(h',h,k') ← a(h',h,k') ∪ X_c        % add assignment to previous transition
34: end while
```

Finalize TA syntax formatting

for all $t(l_i,l_j,k) \in T$ **do** % compile transition guards and updates

$g(l_i,l_j,k) \leftarrow$ '$cl \in g\_cl(l_i,l_j,k)$ /\ $\wedge_{s \in [1,|X|]}$ $x_i \in g\_x(l_i, l_j, k, x_s)$'

$a(l_i,l_j,k) \leftarrow$ '$X_c \leftarrow random(a(l_i,l_j,k))$, $cl \leftarrow 0$' % assign random value in *a*

end for

for all $l_i \in L$ ***do***

$inv(l_i) \leftarrow$ '/\\$_k$ $g(t_{ki})$ /\ $\neg \bigvee_j g(t_{ij})$' % compile location invariants

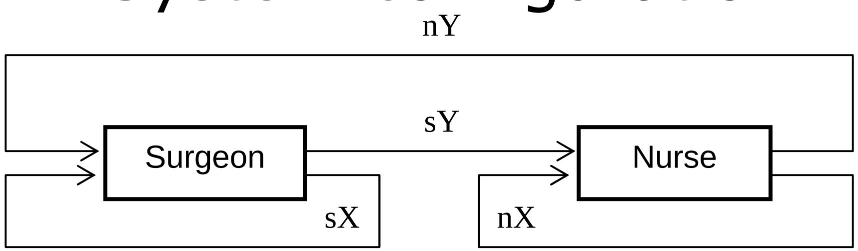
end for

Interval extension operator:

$(.)^{\Updownarrow R} : [x^-, x^+]^{\Updownarrow R} = [x^- - \delta, x^+ + \delta]$, where $\delta = R - (x^+ - x^-)$

Learning example (1)

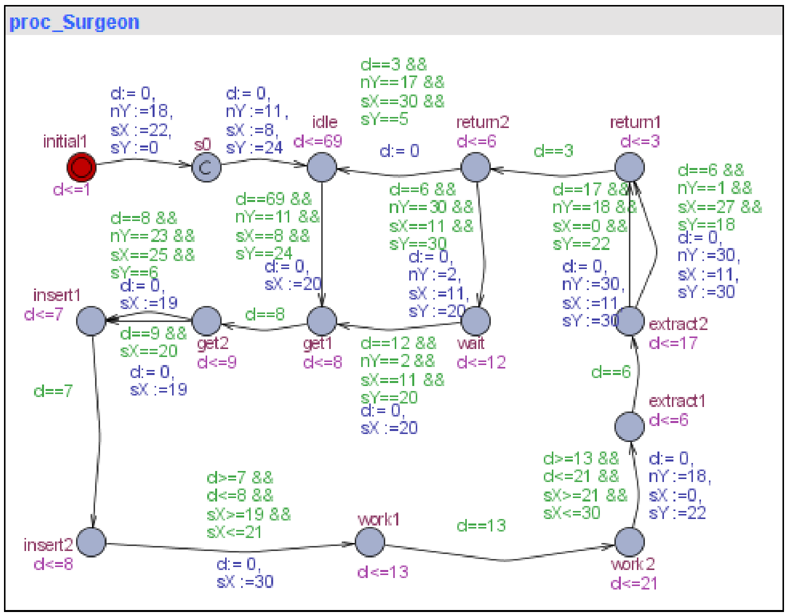
- Given
  - Observation sequence E =
  - System configuration



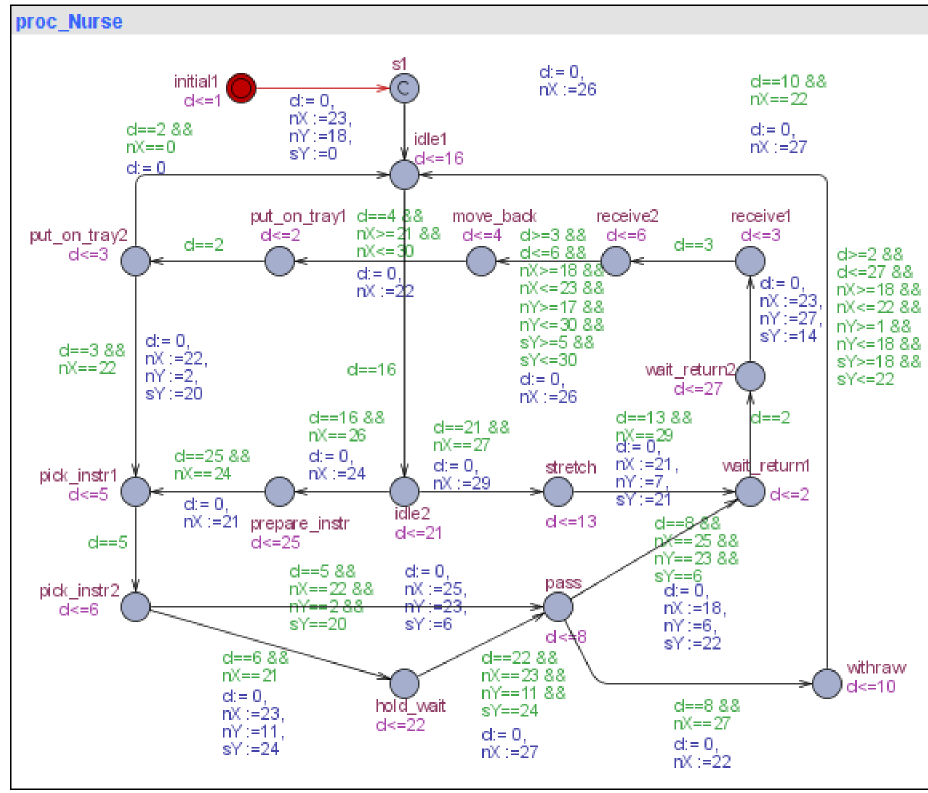
	$I_2^{Surg}\|O_2^{Surg}$	$I_1^{Nurse}\|O_1^{Surg}$	$I_2^{Nurse}\|O_2^{Nurse}$	$I_1^{Surg}\|O_1^{Nurse}$	Action	
Time	sX	sY	nX	nY	Surgeon	Nurse
1	123	52	214	64	idle	idle
17	$	76	237	34	$	prepare_instr
42	$	93	222	85	$	pick_instr
48	$	57	191	55	$	hold_wait
70	81	123	212	46	get	pass
78	$	132	245	72	$	withraw
79	118	85	$	26	insert	$
86	116	73	$	85	work	$
88	$	73	202	66	$	idle
107	121	59	$	44	extract	$
109	$	77	244	88	$	stretch
122	$	86	259	35	$	wait_return
124	59	116	199	63	return	receive
130	92	139	211	93	wait	move_back
134	$	75	194	55	$	put_on_tray
137	$	104	201	33	$	pick_instr
142	92	110	201	26	get	pass
150	133	68	230	76	insert	wait_return
158	121	76	$	55	work	$
171	146	63	$	27	extract	$
177	138	105	170	22	return	receive
180	147	66	169	62	idle	move_back
184	$	124	268	90	$	put_on_tray
186	$	73	20	20	$	idle

  - Rescaling operator $\mathcal{R}$ with region [0,30] for all x_i
  - Granularity of the quotient space $\boldsymbol{X}/_{\sim}$:   2

Learning example (2)



||



Does XTA exhibit the same behavior as the traces observed?

- Question 1: How to choose the equivalence relation "~" do define a feasible quotient space?

- Question 2: How to choose the equivalence relation to compare traces $TTr(XTA)$ and $TTr(Obs)$?

$$TTr(XTA) = \mathcal{R}(TTr(Obs)) /_{\sim} \quad ?$$

How to choose the equivalence relation "$\sim$" do define a feasible quotient space?

- Granularity parameter γ_i defines the maximum length of an interval $[x^-_i, x^+_i)$, of equivalent (regarding $\sim$) values of x_i where $x^-_i, x^+_i \in X_i$ for all X_i ($i = [1,n]$).
- Partitioning of $\mathit{dom}\, X_i$ ($i = [1,n]$) to intervals (see line 16 of the algorithm) is implemented using interval expansion operator $(.)^{\Updownarrow R}$:

$$[x^-, x^+]^{\Updownarrow R} = [x^- - \delta, x^+ + \delta], \text{ where } \delta = R - (x^+ - x^-)$$

On Question 2:

- Possible candidates:
  - Equivalence of languages?
  - Simulation relation?
  - Conformace relation?
  - ...?

How to choose the equivalence relation to compare languages? Nerode's right congruence.

- Given a language $\mathcal{L}(\mathcal{A})$, we say that two words $u, v \in \Sigma^*$ are *equivalent*, written as $u \equiv_{\mathcal{L}(\mathcal{A})} v$, if, $\forall\, w \in \Sigma^*$: $uw \in \mathcal{L}(\mathcal{A})$ iff $vw \in \mathcal{L}(\mathcal{A})$.

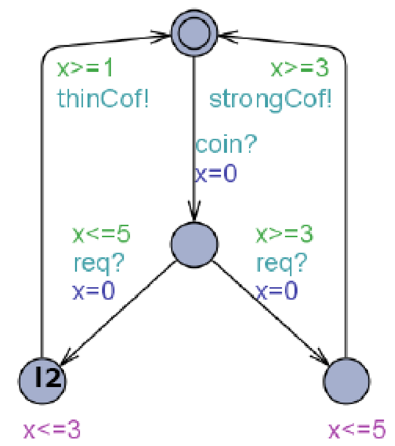
- $\equiv_{\mathcal{L}(\mathcal{A})} \subseteq \Sigma^* \times \Sigma^*$ is a right congruence, i.e., it is an equivalence relation that satisfies

  $$\forall\, w \in \Sigma^*:\ u \equiv_{\mathcal{L}(\mathcal{A})} v \Rightarrow uw \equiv_{\mathcal{L}(\mathcal{A})} vw.$$

- We denote the equivalence class of a word w wrt. $\equiv_{\mathcal{L}(\mathcal{A})}$ by $[w]_{\mathcal{L}(\mathcal{A})}$ or just $[w]$

- A language $\mathcal{L}(\mathcal{A})$ is regular iff the number of equivalence classes of Σ^* with respect to $\equiv_{\mathcal{L}(\mathcal{A})}$ is finite.

Timed Conformance



- Derived from Tretman's IOCO

- Let **I, S** be timed I/O LTS, P a set of states
- **TTr**(P): the set of *timed traces* from P
  - eg.: σ = coin?.**5**.req?.**2**.thinCoffee!.**9**.coin?
- **Out**(P **after** σ) = possible ***outputs*** and ***delays*** after σ
  - eg. out ({l2,x=1}): {thinCoffee, **0...2**}

- **I rt-ioco S =def**
  - **∀σ ∈ TTr(S): Out(I after σ) ⊆ Out(S after σ)**
  - **TTr(I) ⊆ TTr(s)** *if s and I are input enabled*

- **Intuition**
  - **no illegal output is produced and**
  - **required output is produced (at right time)**

See also [Krichen&Tripakis, Khoumsi]

Conclusions

- Proposed XTA learning makes the on-line synthesis of model-based planning controllers for HA robots feasible.
- Other aspects:
  - *learning is incremental*, i.e., pre-existing knowledge about the agent's behaviour can be re-used;
  - *formal semantics* of XTA models - functional correctness and performance can be verified on the model before used for planner synthesis;
  - *adjustable level of abstraction* of the model generated.