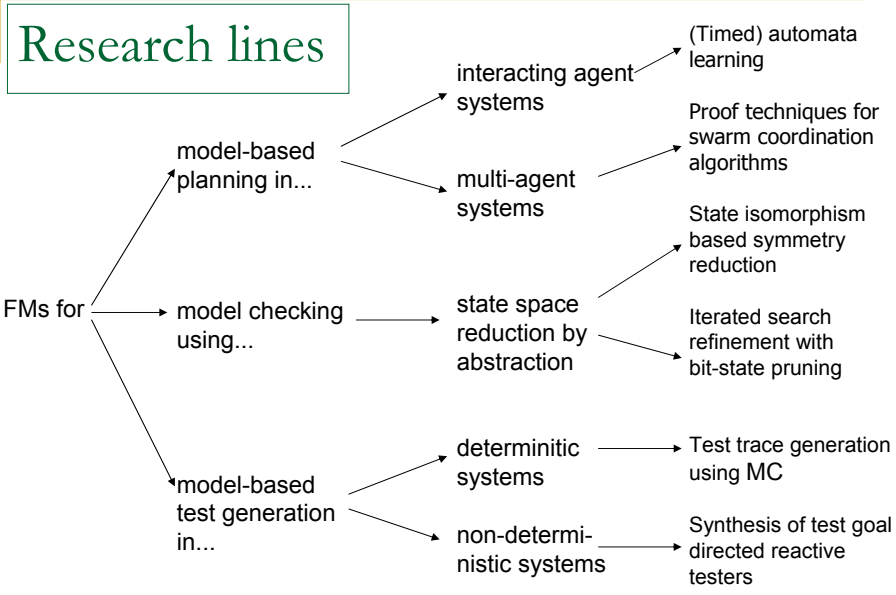
CREDES related research at the Dept. of Computer Science TUT

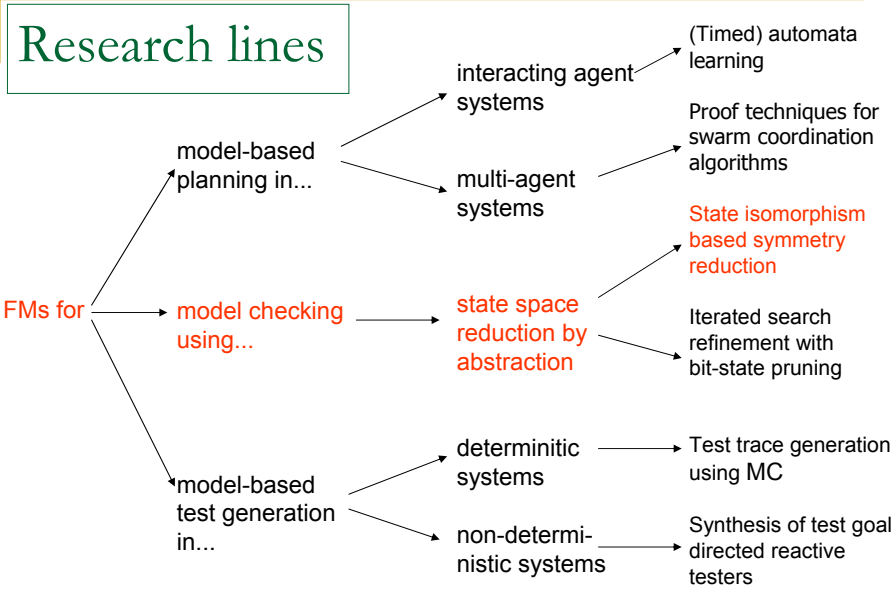
Jüri Vain

CREDES Kick-off MeetingTallinn, June 05, 2009

Research lines



Research lines

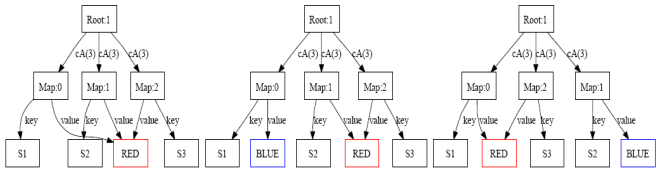


State isomorphism based symmetry reduction

- Solution
  - Given a model in terms of unordered data structures, like sets and maps, and objects:
    - Construct the state graphs of the states at run time and
    - Use the computation of graph isomorphism to determine whether a state with similar structure has been seen.
- Results developed with Margus Veanes and Colin Campell and published at FORTE 2007 (June 2007) and in the PhD thesis of J. Ernits (Nov. 2007)

State isomorphism based symmetry reduction

- Problem
  - Exploit symmetries to reduce the effects of state space explosion in explicit state reachability

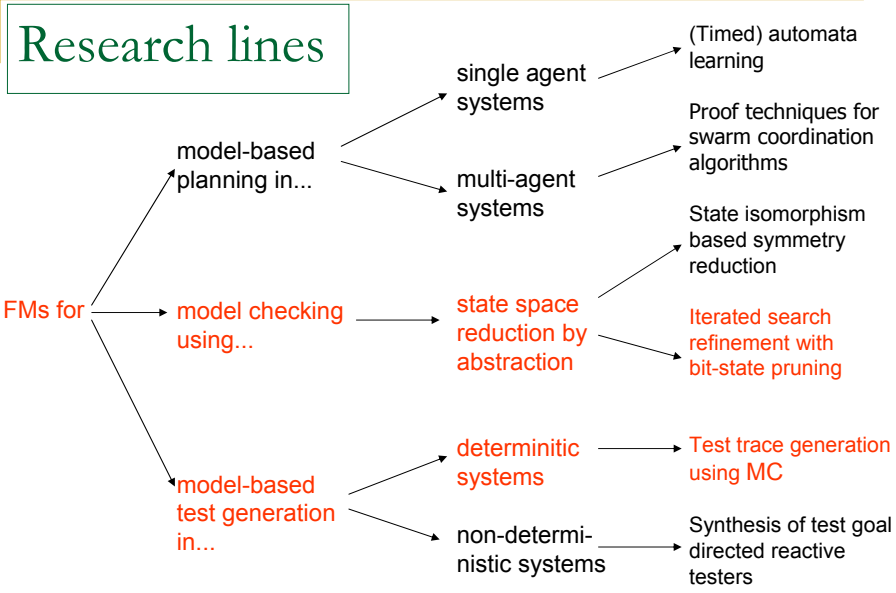


State isomorphism based symmetry reduction

- Applications:
  - Method is implemented in model-testing toolkit called NModel

    (by Margus Veanes et al, MS Research Redmond)
  - There is similar implementation developed independently by Arend Rensink in the GROOVE project

Research lines



Iterated search refinement with bit-state pruning

- Problem
  - Calculate reachability in models that
    - Run out of resources without producing results with known methods
    - Produce too long traces to the error/desired state with known methods
  - Such reachability problems can be used for solving problems related to
    - Scheduling
    - Hardware synthesis
    - Offline test generation

Iterated search refinement with bit-state pruning

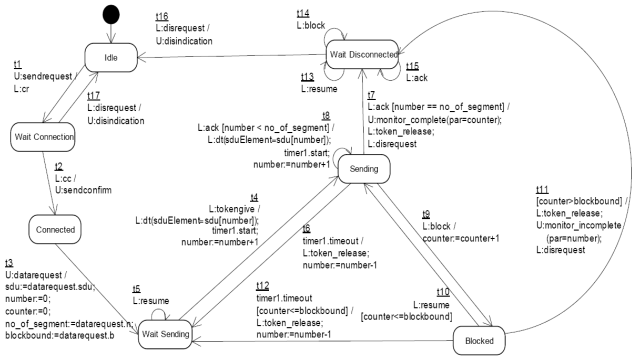
- Solution
  - Combine two well known methods in a new way:
    - Iterated Search Refinement, and
    - Bitstate hashing
  - The key idea is to use collisions in bitstate hash table to randomly filter the search space.
    - Each iteration requires very small amounts of memory
    - A lot of prefixes of paths are covered multiple times, but
    - Provides good results in several examples.
- Results published in the PhD thesis of J. Ernits (Nov. 2007)

Iterated search refinement with bit-state pruning

- Practical applications
  - 2 well researched case studies:
    - Synthesis of a memory arbiter for a radar memory case study (J. Ernits, 2005)
    - Calculating offline test sequences for model-based testing (J. Ernits, A. Kull, K. Raiend, J. Vain, 2006)
    - The method cannot be used for disproving the reachability, but provides a new alternative for finding a witness trace.

Pre-set test trace generation using MC

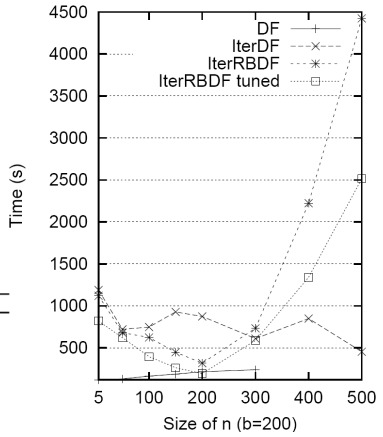
Case study: modified INRES protocol



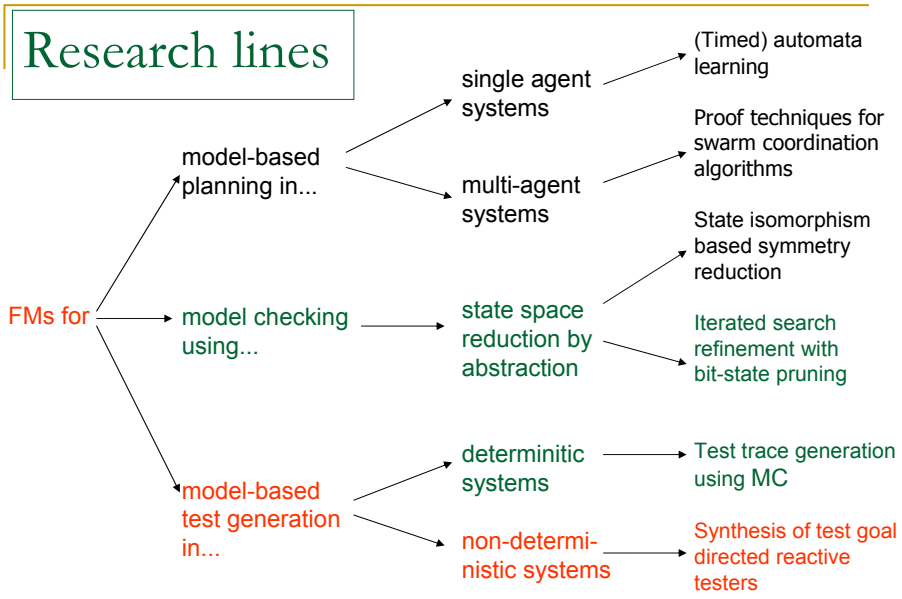
Pre-set test trace generation using MC

- Results
  - Time spent for finding sequences for the 2-switch coverage criterion
  - Experiments made with Uppaal Cora

Abbreviation	ISR	Search order	Trace	Guiding
DF	-	depth first	some	-
IterDF	first result	depth first	some	-
IterRBDF	first result	random best depth first	best	uniform
IterRBDF tuned	first result	random best depth first	best	tuned



Research lines



Synthesis of test goal directed reactive testers

- Problem:
  - Given a
    - Non-deterministic EFSM M_{SUT} of System Under Test
    - testing goal in terms of a set ***Trp*** of M_{SUT} transitions
  - Find EFSM M_{TSR} s.t.
    - M_{TSR} is I/O compliant with M_{SUT}
    - any trace of $M_{SUT} \| M_{TSR}$ includes labels of all transitions in ***Trp***
    - M_{TSR} chooses a transition from those labelled with current input from M_{SUT} that maximizes some gain function *G*.

Synthesis of test goal directed reactive testers

- Solution:
  - Algorithm of constructing M_{TSR}
- Complexity:
  - For all controllable transitions of the M_{TSR} the upper bound of the complexity of the computations of the gain functions is $O(|E_{SUT}|^3)$.
  - At runtime each choice by the tester takes $O(|E_{SUT}|^2)$ arithmetic operations to evaluate the gain functions
- (ASE2007, Vain, Raiend, Kull, Ernits)

Synthesis of test goal directed reactive testers

- Experimental results

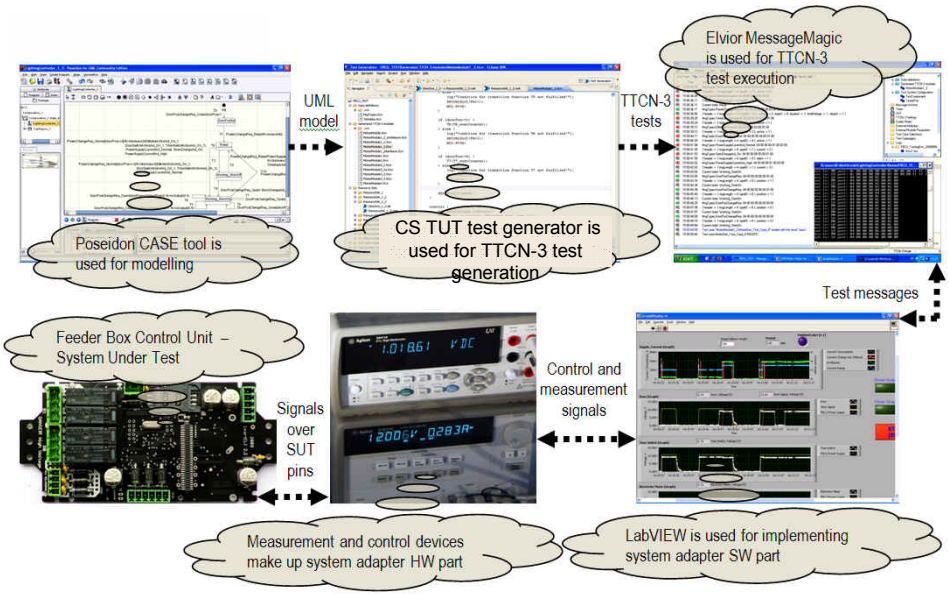
All Transition Test Purpose

Algorithm of the tester	Model 1 (8 trans.)	Model 2 (16 trans.)	Model 3 (32 trans.)
Random choice	56 ± 36	295 ± 130	1597 ± 1000
Anti-ant	21 ± 4	53 ± 13	218 ± 81
Reactive planner	17 ± 3	37 ± 6	80 ± 10

One Transition Test Purpose

Algorithm of the tester	Model 1 (8 trans.)	Model 2 (16 trans.)	Model 3 (32 trans.)
Random choice	34 ± 35	120 ± 114	699 ± 719
Anti-ant	14 ± 7	36 ± 19	140 ± 70
Reactive planner	5 ± 2	8 ± 3	11 ± 3

State-of-the-art



Research lines

- FMs for
  - model based planning in...
    - single agent systems
      - (Timed) automata learning
    - multi-agent systems
      - Proof techniques for swarm coordination algorithms
  - model checking using...
    - state space reduction by abstraction
      - State isomorphism based symmetry reduction
      - Iterated search refinement with bit-state pruning
  - model based test generation in...
    - deterministic systems
      - Test trace generation using MC
    - non-determi-nistic systems
      - Synthesis of test goal directed reactive testers

Proof techniques for swarm coordination algorithms

- Problem:
  - proving properties of swarm (distributed) coordination algorithms
  - self-stabilization, covergence, ... can be reduced to reachability analysis
  - reachability analysis collides with scalability barrier

Proof techniques for swarm coordination algorithms (ongoing work)

- Some solutions:
  - Pattern based problem encoding provides right level of abstraction (spec. patterns are problem oriented)
  - General principles of proving:
    - Compositional approach combining different techniques of component proofs – MC, deduction,...
    - Global properties by structural induction on the size of swarm/task
      - Base: prove sufficiency of assumptions on a swarm fragment of some tractable size, e.g., using MC with symmetry reduction, (typically the size of fragment can't be trivial either)

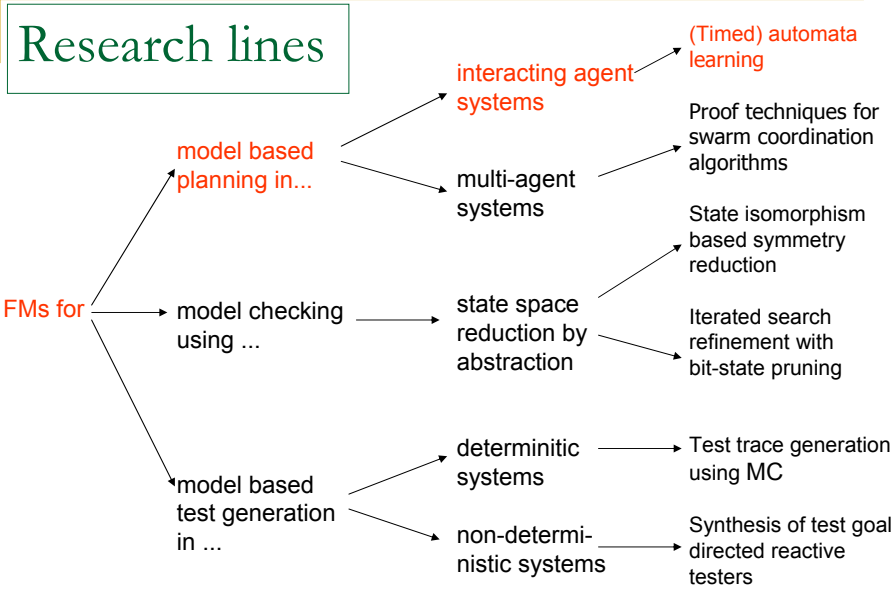
        (see BEC2008, Vain, Kuusik, Tammet, Juurik)
      - Step: Show that the induction step does not violate the assume part for the new fragment defined by the step

Proof techniques for swarm coordination algorithms

- Demo: *dynamic cleaning problem*
  - different zones of the room deteriorate with different rate
  - a swarm of cleaning robots should keep deterioration below treshold
  - robots of the swarm communicate trough *pherromone trace*
- Problem: given a treshold C_{Tresh} and initial value of the deterioration vector $\boldsymbol{S}^{det}(0)$, *s.t.* $S_i^{det}(0) > C_{Tresh}$ for all *i,*
  - prove that
  1) the swarm is always able to reach the state $\boldsymbol{S}^{det}(t)$, *s.t.* $S_i^{det}(t) < C_{Tresh}$ for all *i*,

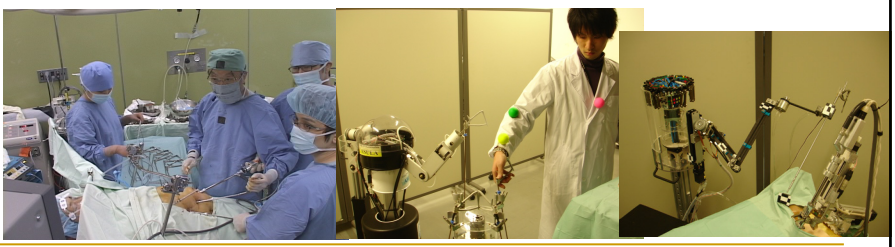
     and
  1) *for all t' > t,* condition $\boldsymbol{S}_t^{det}(t') < C_{Tresh}$ is invariantly true.

Research lines



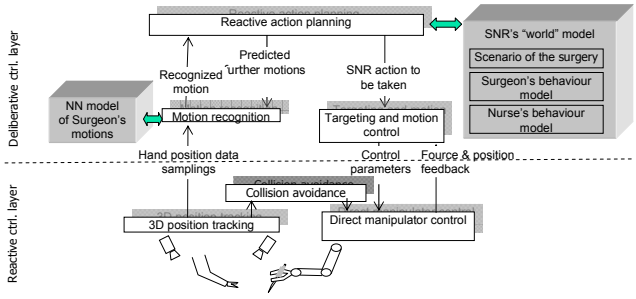
HRI automata learning

- Problem: Planner synthesis for a human adaptive Scrub Nurse Robot (SNR)
  - human adaptive = on-line learning from **H-H**/**H-R** interactions
  - learning = constructing/modifying a human motion model



(Timed) automata learning

- Context: SNR control architecture



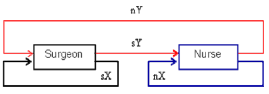
(Timed) automata learning

- Learning algorithm
  - Input:
    - Time-stamped sequences of observations of HRI (set of timed traces $Tr^T(Obs)$)
    - Parameters for
      - state rescaling operator - R
      - distinquishing model observables/controllables - $|_{TAIO}$
      - defining quotient state space
  - Output:
    - Extended (Uppaal version) timed automaton *TA* s.t. $Tr^T(TA) = R(Tr^T(Obs)|_{TAio}) /\sim$ % RTIOCO of timed traces

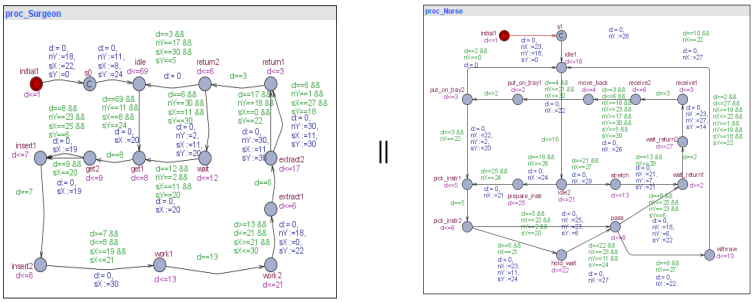
(Vain, Miyawaki, ICCAS-SICE2009, Fukuoka, Japan)

(Timed) automata learning (example)

Event		TS	Surgeon				Nurse				Notations:
Sur	Nur		X_s	X^w_s	Y_s	Y^w_s	X_n	X^w_n	Y_n	Y^w_n	X_s, Y_s, X_n, Y_n – x, y-coordinates of surgeon's and nurse's wrist
σ^S_0	σ^N_0	1	123	22	52	0	214	23	64	11	$X^w_s, Y^w_s, X^w_n, Y^w_n$ - normalized in interval [0.30] X_s, Y_s, X_n, Y_n coordinates
-	σ^N_1	17	-	-	76	-	237	26	34	-	
-	σ^N_2	42	-	-	93	-	222	24	85	-	
-	σ^N_3	48	-	-	57	-	191	21	55	-	
σ^S_1	σ^N_4	70	81	8	123	24	212	23	46	11	*TS* – switching event time stamp
-	σ^N_5	78	-	-	132	-	245	27	72	-	**Switching events of Nurse's Gestures:**
σ^S_2	-	79	118	20	85	6	-	-	26	23	σ^N_0 – idle
σ^S_3	-	86	116	19	73	-	-	-	85	-	σ^N_1 – prepare instrument
-	σ^N_6	88	-	-	73	-	202	22	66	-	σ^N_2 – picking up an instrument
σ^S_4	-	107	121	21	59	-	-	-	44	-	σ^N_3 – holding the instrument & waiting
-	σ^N_7	109	-	-	77	-	244	27	88	-	σ^N_4 – passing the instrument
-	σ^N_8	122	-	-	86	-	259	29	35	-	σ^N_5 – wait returning
σ^S_5	σ^N_9	124	59	0	116	22	199	22	63	18	σ^N_6 – withdrawing hand
σ^S_1	σ^N_9	130	92	11	139	30	211	23	93	30	σ^N_7 – stretching hand
-	σ^N_{10}	134	-	-	75	-	194	21	55	-	σ^N_8 – receiving
-	σ^N_{11}	137	-	-	104	-	201	22	33	-	σ^N_9 – moving back
σ^S_6	σ^N_{12}	142	92	11	110	20	201	22	26	-	σ^N_{10} – putting on the tray.
σ^S_7	σ^N_{13}	150	133	25	65	6	230	25	76	23	**Switching events of Surgeon's gestures:**
σ^S_8	-	158	121	21	76	-	-	-	55	-	σ^S_0 – idle,
σ^S_9	-	171	146	30	63	-	-	-	27	-	σ^S_1 – receiving an instrument;
σ^S_4	σ^N_{14}	177	138	27	105	18	170	18	22	-	σ^S_2 – inserting instrument;
σ^S_1	σ^N_{15}	180	147	30	66	5	169	18	62	17	σ^S_3 – working;
-	σ^N_{16}	184	-	-	124	-	268	30	90	-	σ^S_4 – waiting for an instrument;
-	σ^N_{17}	186	-	-	73	-	20	0	20	-	σ^S_5 – extracting from trocar cannula;
											σ^S_6 - returning the instrument.
											–working with i^{th} instrument, -- working with $i+1^{th}$ instrument



Interaction model constructed



MB Test &Verification

SoA	Beyond SoA
• MB test generation:	
M construction off-line	⇒ on-line *learning* of SUT models
• Testing non-deterministic systems:	
Random walk, anti-ant, ..	⇒ goal-oriented on-line *planning* testers: RPT
• Test data construction:	
off-line constraint solving	⇒ on-the-fly constraint *construction* & solving
• Distributed testing	
with centralized control	⇒ with *fully distributed* control (*coordination*&*synchronization*)

- Thank You!