# TDDE13 Multiagent Systems

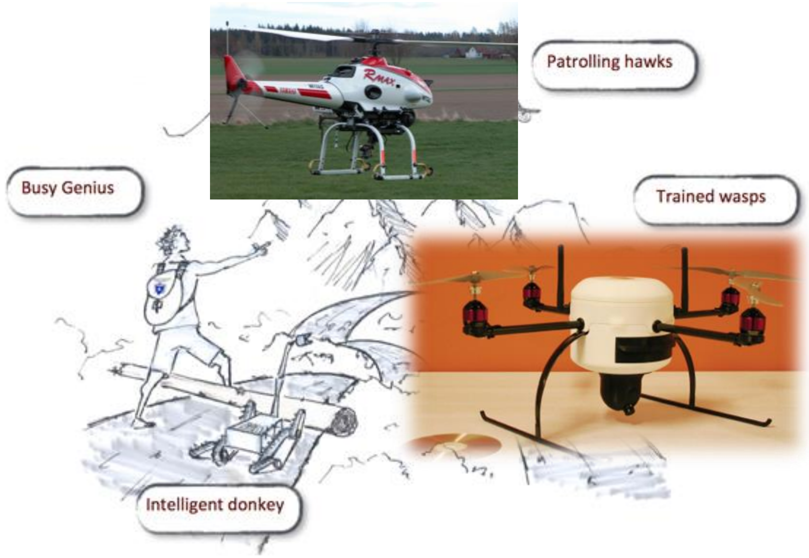
Fredrik Heintz, Dept. of Computer Science,
Linköping University, Sweden

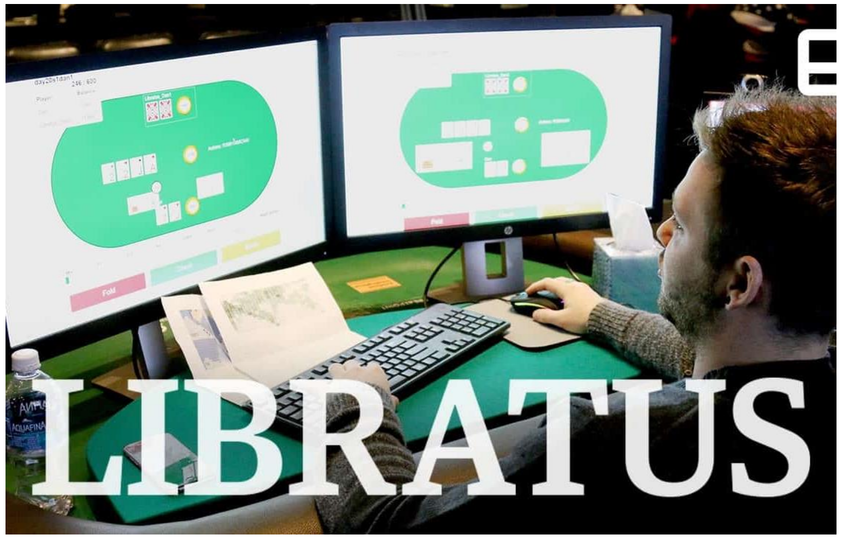
fredrik.heintz@liu.se

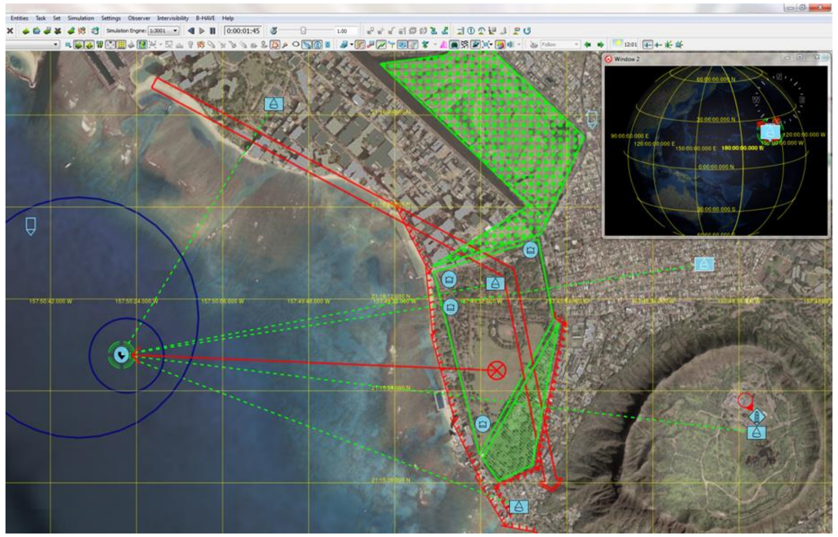
@FredrikHeintz

LINKÖPINGS
UNIVERSITET

# Multiagent systems







## Learning Outcomes

- The overall aim of the course is to give an overview of multiagent systems and in-depth knowledge of some areas of multiagent systems. After the course students should be able to:
  - List and explain important problems and techniques in the area of multiagent systems.
  - Explain how central algorithms in the area of multiagent systems work.
  - Be able to implement some central algorithm in the area of multiagent systems.
  - Evaluate and apply different game theoretic approaches.
  - Design and use auctions for allocating resources in a multiagent system.
  - Model relevant aspects of multiagent system decision making using markov decision processes and logics.

## Lectures and Seminars Part I

- The course material for week 1-3 are the following lectures from Game Theory Online:
  - Game Theory I - Week 1 (Introduction)
  - Game Theory I - Week 2 (Mixed-strategy Nash equilibria)
  - Game Theory I - Week 3 (Alternate solution concepts)
  - Game Theory I - Week 4 (Extensive-Form Games)
  - Game Theory I - Week 5 (Repeated Games)
  - Game Theory I - Week 6 (Bayesian Games)
  - Game Theory I - Week 7 (Coalitional Games)
- and the following pre-recorded lectures
  - Cooperative game theory
  - Coalition formation and centralized algorithms
- *Nov 19, Lab session: Centralized coordination algorithms*
- *Nov 20, Seminar: Game theory (**exercise set 1**)*
- Dec 6, **Deadline**: Lab 1 accepted Kattis submissions and report submitted

# Lectures and Seminars Part II

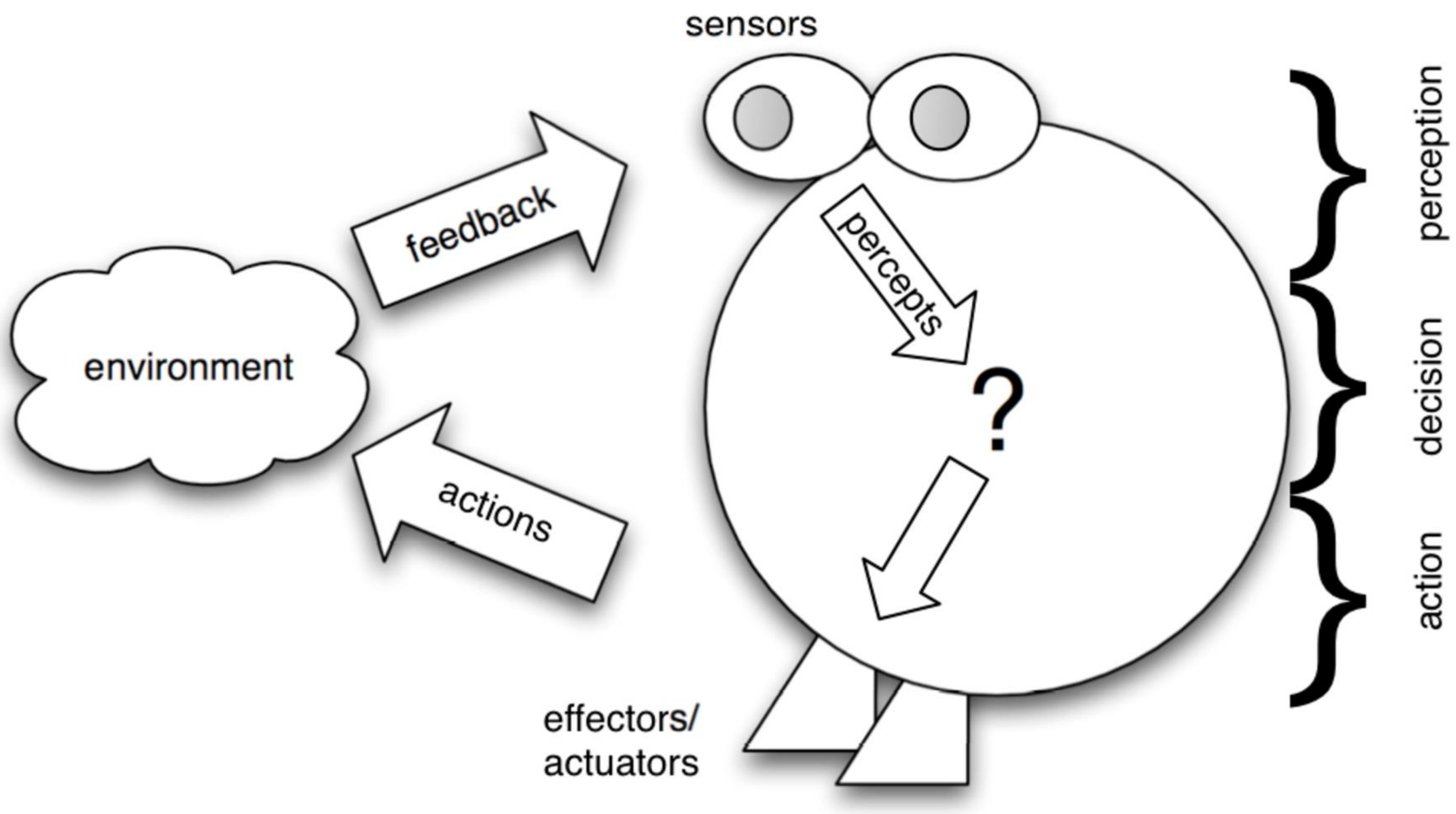
- The course material for week 4 and 5 are the following lectures from Game Theory Online:
  - Game Theory II - Week 1 (Social Choice)
  - Game Theory II - Week 2 (Mechanism Design)
  - Game Theory II - Week 3 (VCG)
  - Game Theory II - Week 4 (Auctions)
- and the following pre-recorded lectures
  - Multi-agent learning
- *Dec 3, Lab session: Multi-agent reinforcement learning*
- *Dec 4, Seminar: Social choice, mechanism design and auctions (**exercise set 2**)*
- Dec 4, **Deadline**: Approved choice of subject for the individual report
- *Dec 7-8, Discuss individual reports individually with Fredrik Präntare*
- *Dec 17, Seminar: Student presentations on individual report*
- Dec 13, **Deadline**: Lab 2 report submitted
- Dec 22, **Deadline**: Individual report submitted

## Examination

- **LAB 2hp**
  - Centralized Coordination Algorithms
  - Multiagent Reinforcement Learning
- **UPG 4hp**
  - 0-17 points grade U; 18-23 points grade 3;
    24-26 points grade 4; 27-28 points grade 5
  - **[7 points]** Assignment Set 1: Agents and Game Theory
  - **[7 points]** Assignment Set 2: Mechanism Design, Social Choice, and Coalitional Game Theory
  - **[14 points]** Individual written report + presentation at seminar

## Intelligent Agents

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

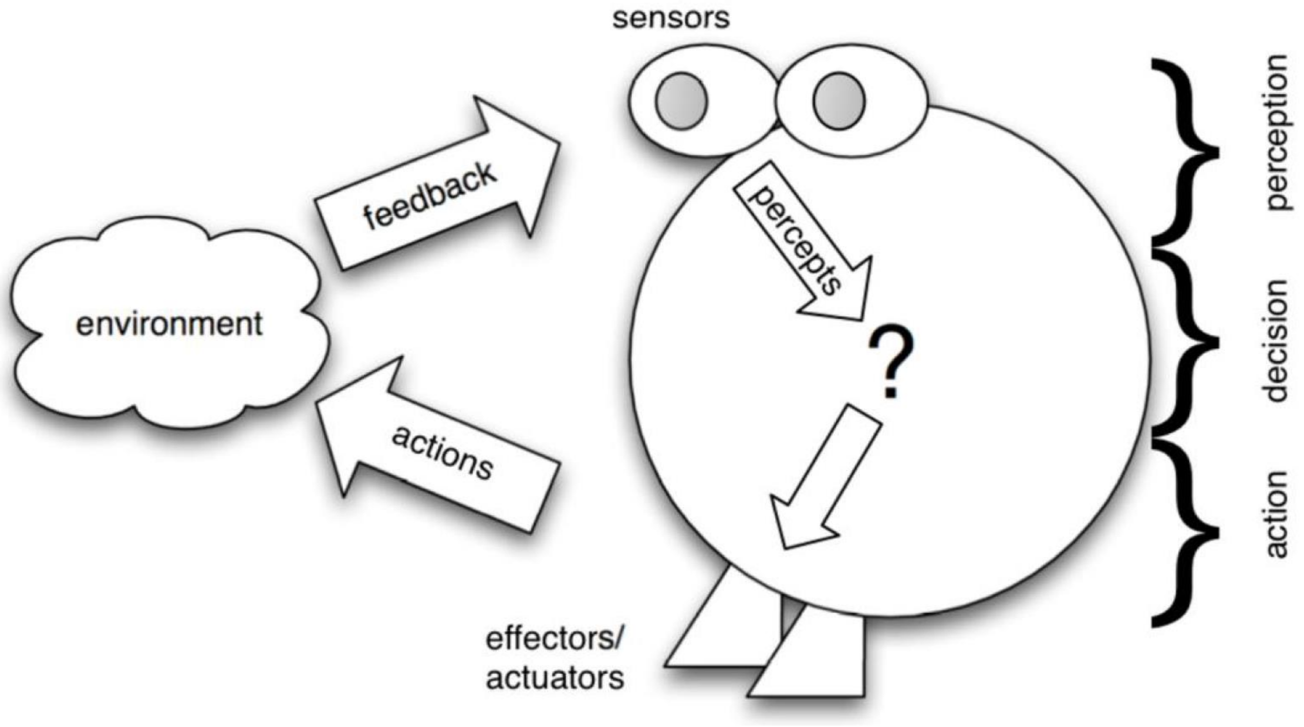


# Intelligent Agents

Classical AI
Modern AI

**AI Technology areas**

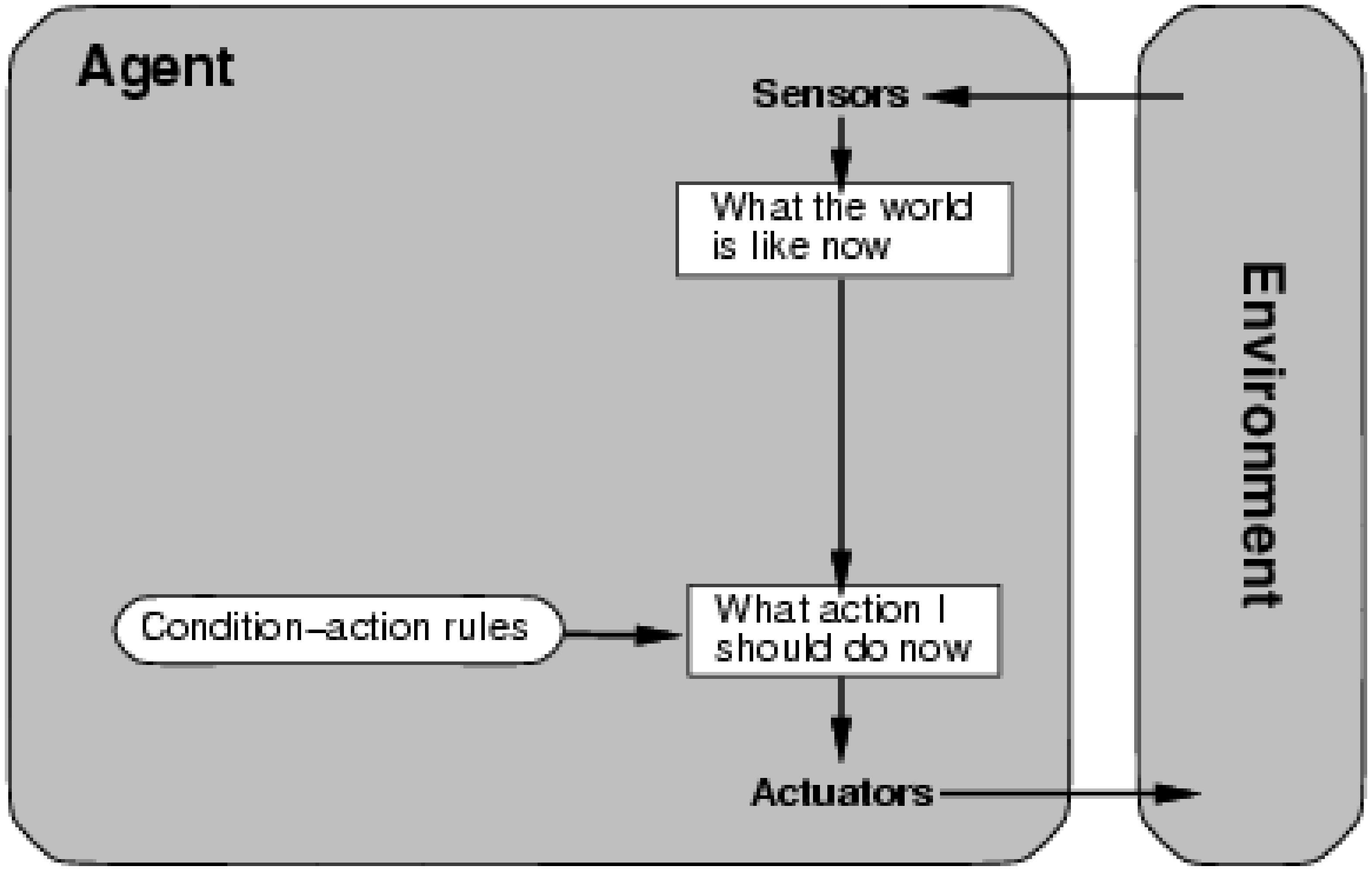
- Perception
- Learning
- Knowledge representation and reasoning
- Planning and decision making
- Control



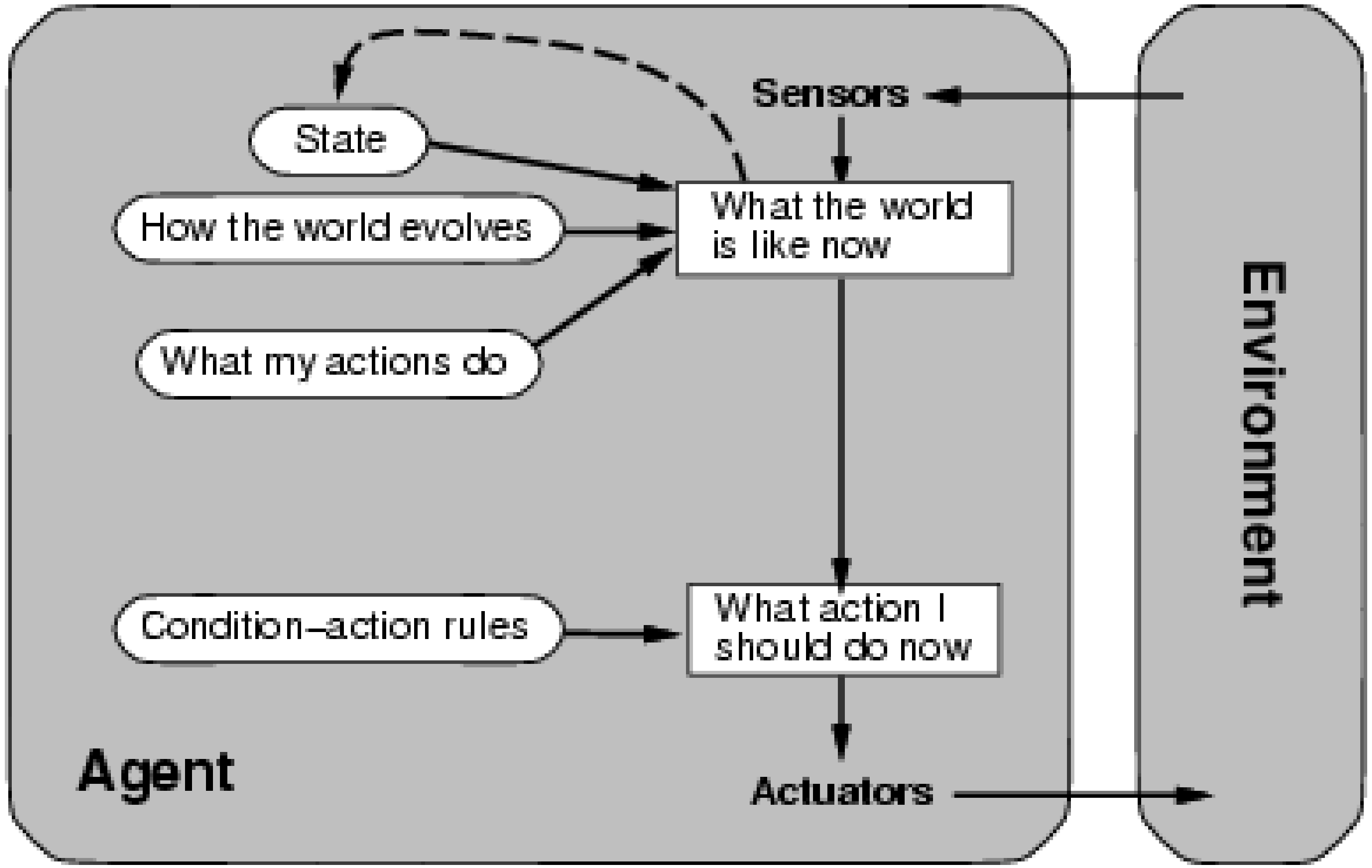
## Agent types

- Four basic types in order of increasing generality:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

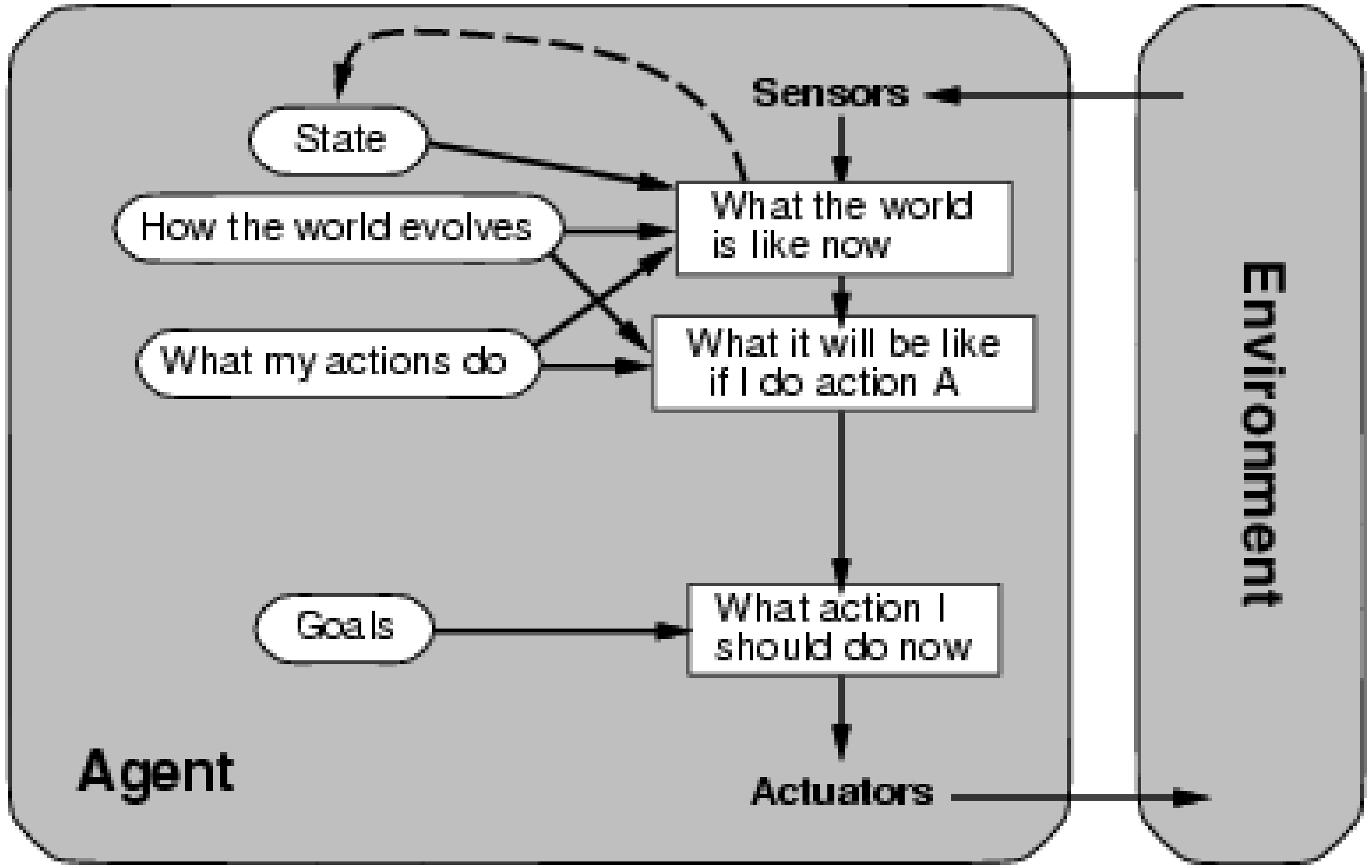
## Simple reflex agents



# Model-based reflex agents

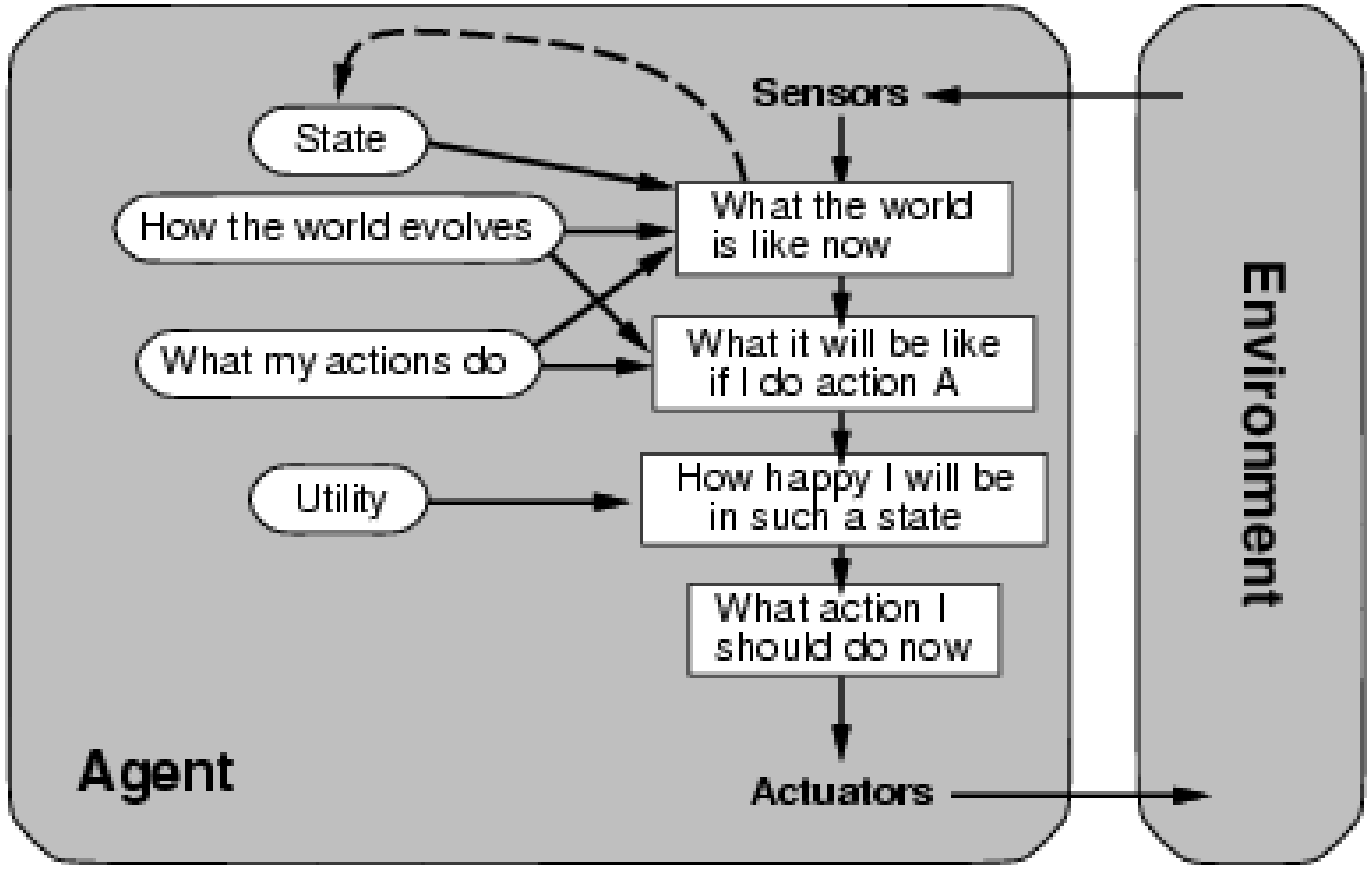


## Goal-based agents

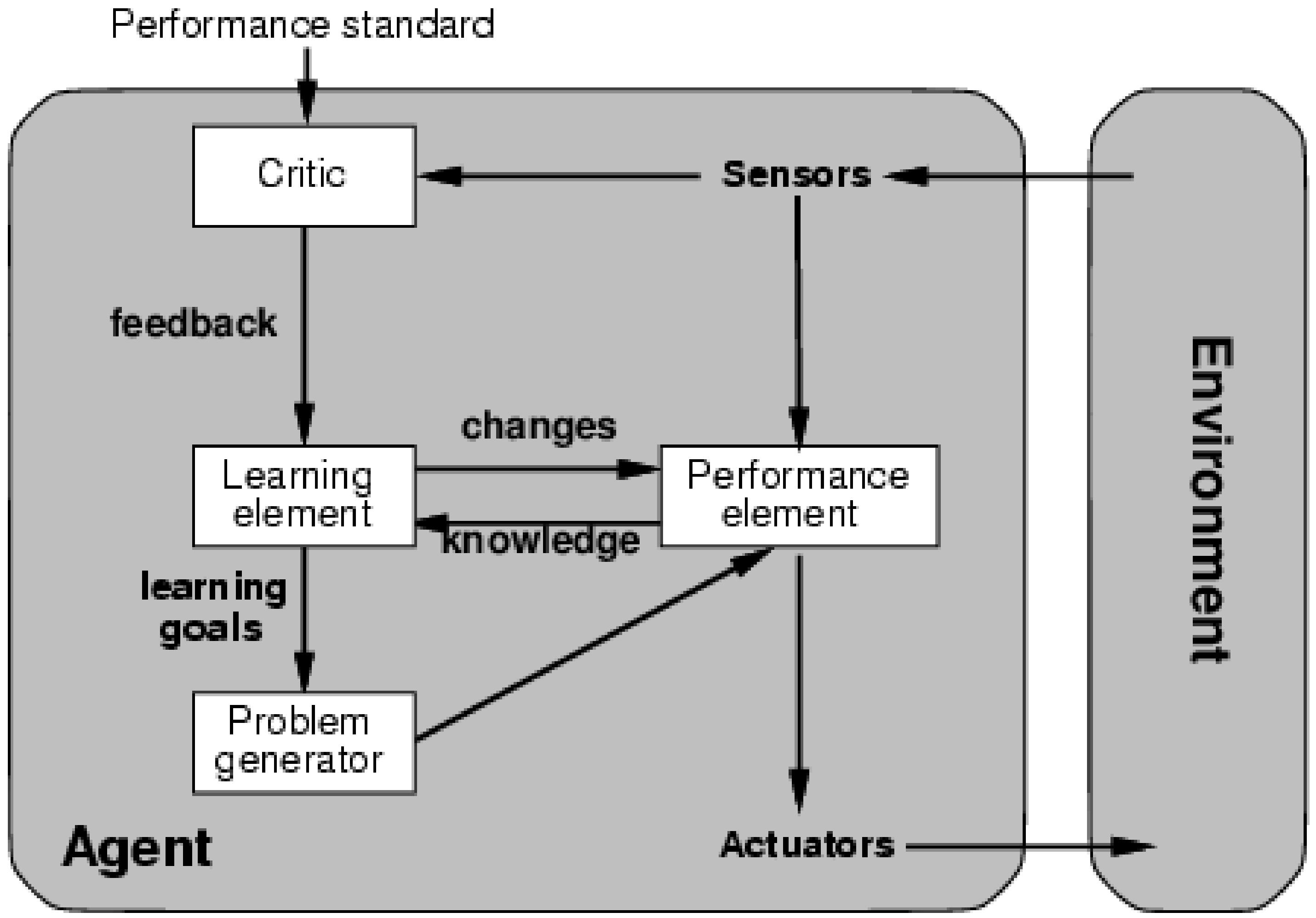

Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

LINKÖPINGS UNIVERSITET

## Utility-based agents


Sensors
State
How the world evolves
What my actions do
Utility
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

## Learning agents



## Agent Architectures Summary

- Originally (1956-1985), pretty much all agents designed within AI were *symbolic reasoning* agents
- Its purest expression proposes that agents use *explicit logical reasoning* in order to decide what to do
- Problems with symbolic reasoning led to a reaction against this — the so-called *reactive agents* movement, 1985–present
- From 1990-present, a number of alternatives proposed: *hybrid architectures*, which attempt to combine the best of reasoning and reactive architectures

# Deliberative Architectures

## Properties

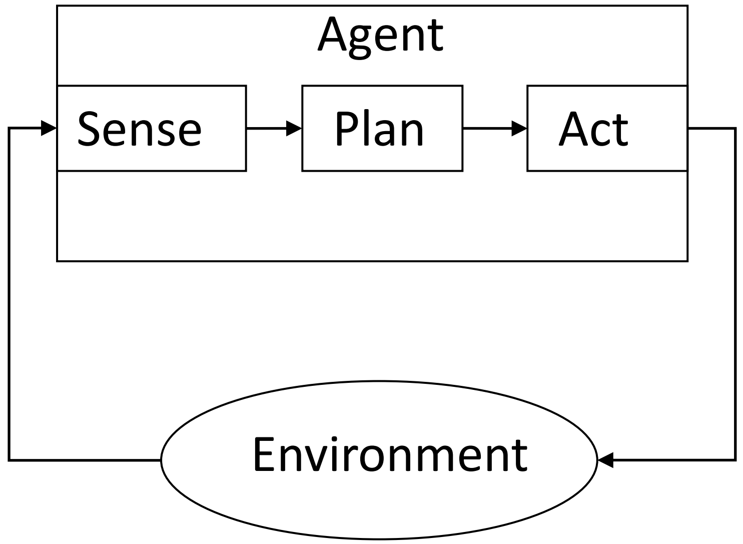
- Internal state
  (using symbolic representation)
- Search-based decision making
- Goal directed

## Benefits

- Nice and clear (logics) semantics
- Easy to analyze by proving properties

## Problems

- Can’t react in a timely manner to events that
  requires immediate actions. Intractable algorithms.
- Hard to create a symbolic representation from
  continuous sensor data. The anchoring problem.



# Reactive Agent Architectures

**Properties**

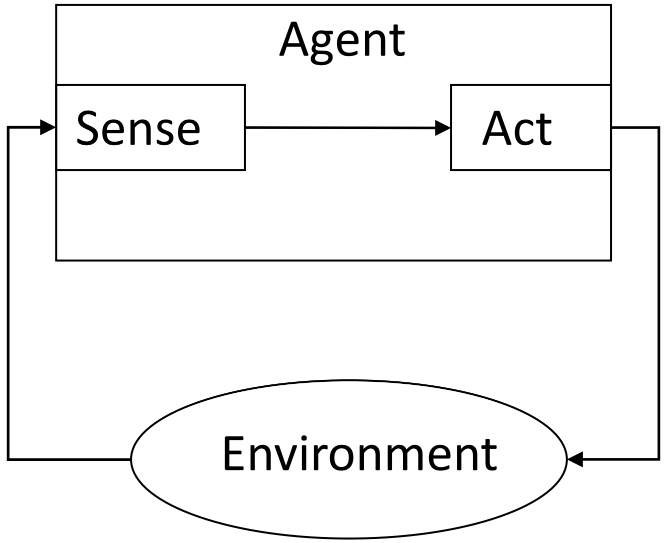
- No explicit world model
- Rule-based decision making

**Benefits**

- Efficient
- Robust

**Problems**

- The local environment must contain enough information to make a decision.
- Easy to build small agents, hard to build agents with many behaviors or rules. Emergent behavior.



## Hybrid Agent Architectures

**Properties**

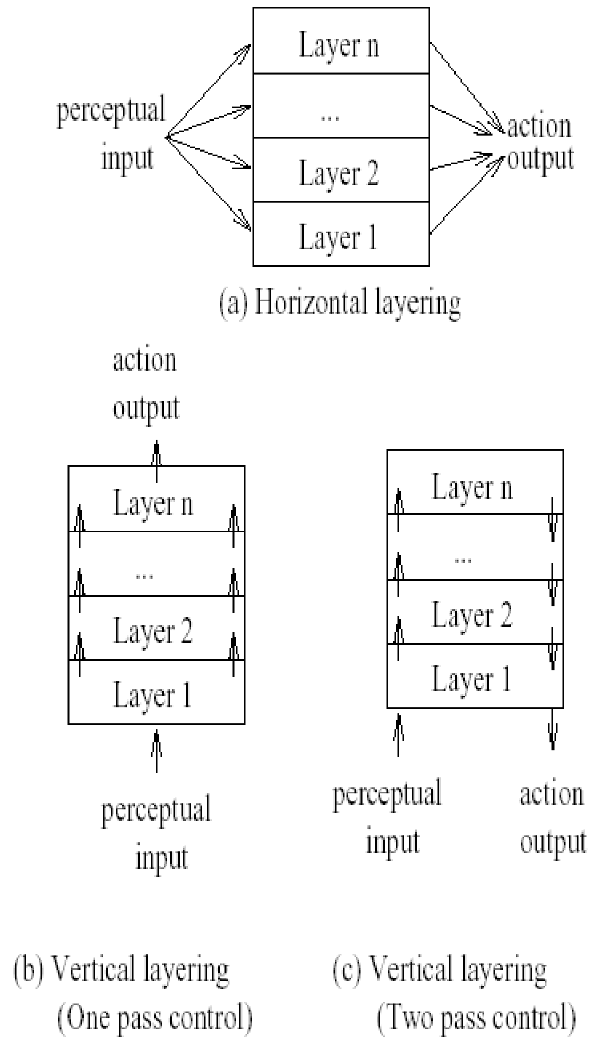
- Tries to combine the good parts of both reactive and deliberative architectures.
- Usually layered architectures.

**Benefits**

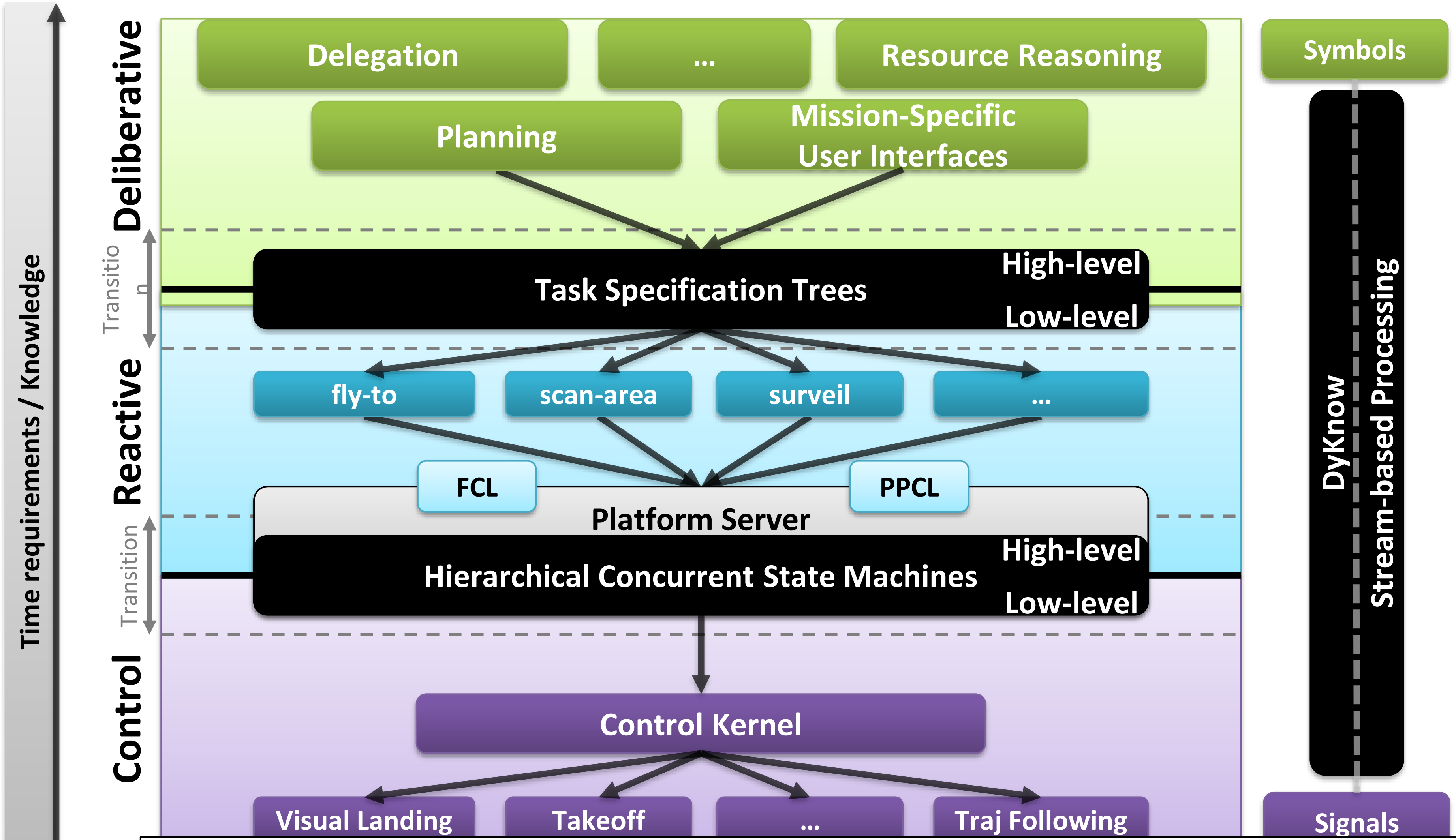
- Attacks the problem on different abstraction levels.
- Has the benefits of both architecture types.

**Problems**

- Hard do combine the different parts.



# HDRC3: A Distributed Hybrid Deliberative/Reactive Architecture for Autonomous Systems



P. Doherty, J. Kvarnström, M. Wzorek, P. Rudol, F. Heintz and G. Conte. 2014.
**HDRC3 - A Distributed Hybrid Deliberative/Reactive Architecture for Unmanned Aircraft Systems**.
In K. Valavanis, G. Vachtsevanos, editors, Handbook of Unmanned Aerial Vehicles, pages 849–952.

# Human and Computational Thinking

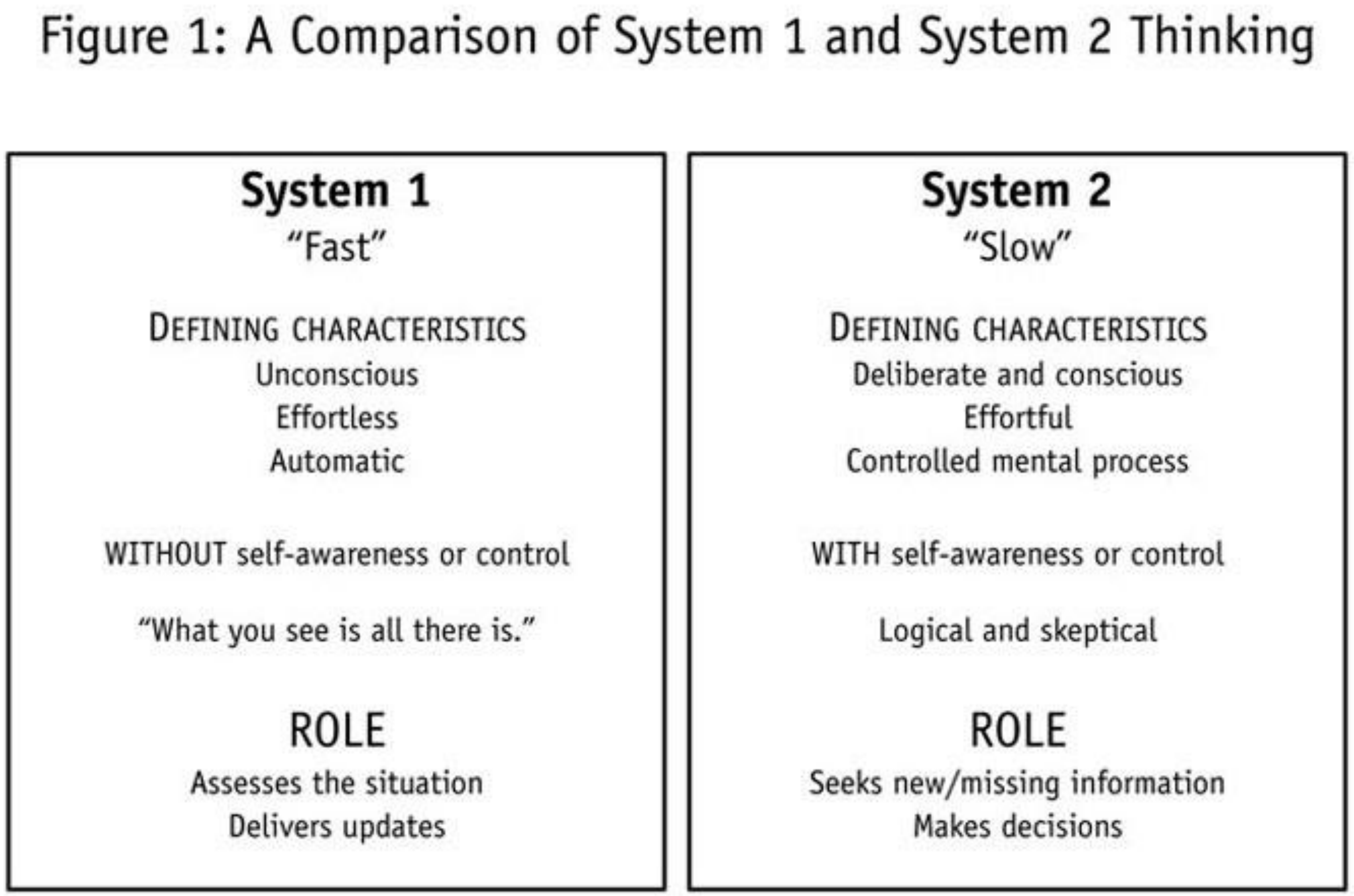
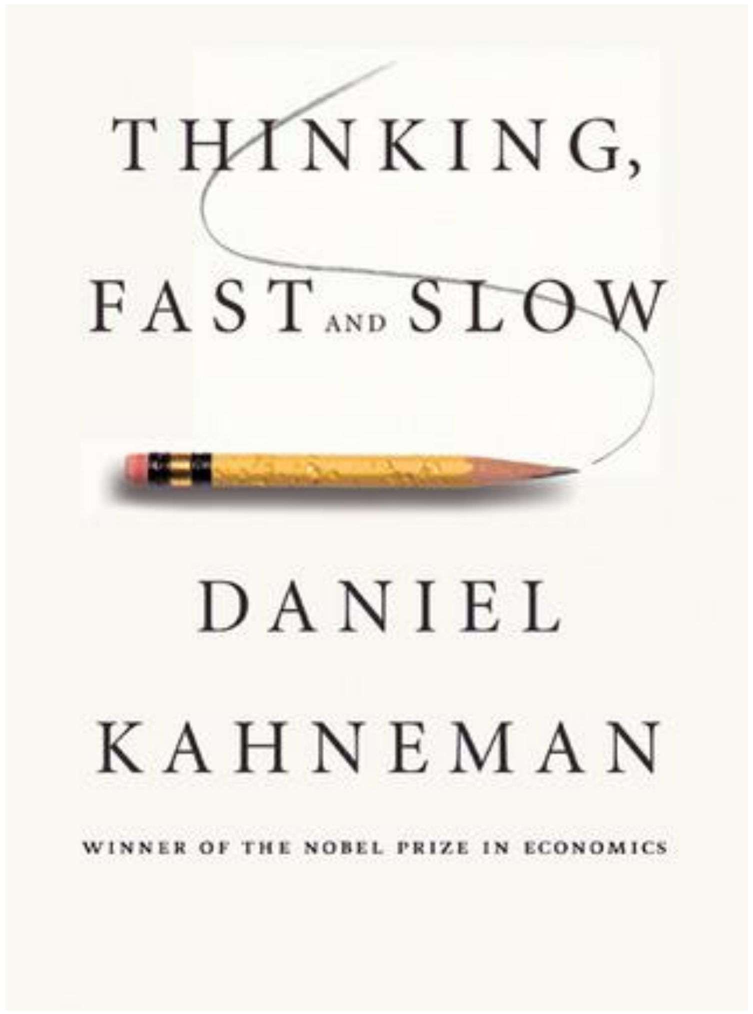


Figure 1: A Comparison of System 1 and System 2 Thinking

| **System 1** | **System 2** |
|---|---|
| "Fast" | "Slow" |
| DEFINING CHARACTERISTICS | DEFINING CHARACTERISTICS |
| Unconscious | Deliberate and conscious |
| Effortless | Effortful |
| Automatic | Controlled mental process |
| WITHOUT self-awareness or control | WITH self-awareness or control |
| "What you see is all there is." | Logical and skeptical |
| ROLE | ROLE |
| Assesses the situation | Seeks new/missing information |
| Delivers updates | Makes decisions |



LINKÖPINGS UNIVERSITET

SWEDISH ARTIFICIAL INTELLIGENCE SOCIETY

**Pure Logic** **Pure Learning**

- Slow thinking: deliberative, cognitive, model-based, extrapolation
- Amazing achievements until this day
- "*Pure logic is brittle*"
  noise, uncertainty, incomplete knowledge, …

**Pure Logic** **Pure Learning**

- Fast thinking: instinctive, perceptive, model-free, interpolation
- Amazing achievements recently
- "*Pure learning is brittle*"

  bias, algorithmic fairness, interpretability, explainability, adversarial attacks, unknown unknowns, calibration, verification, missing features, missing labels, data efficiency, shift in distribution, general robustness and safety

  fails to incorporate a sensible model of the world

## Multi-Agent Systems

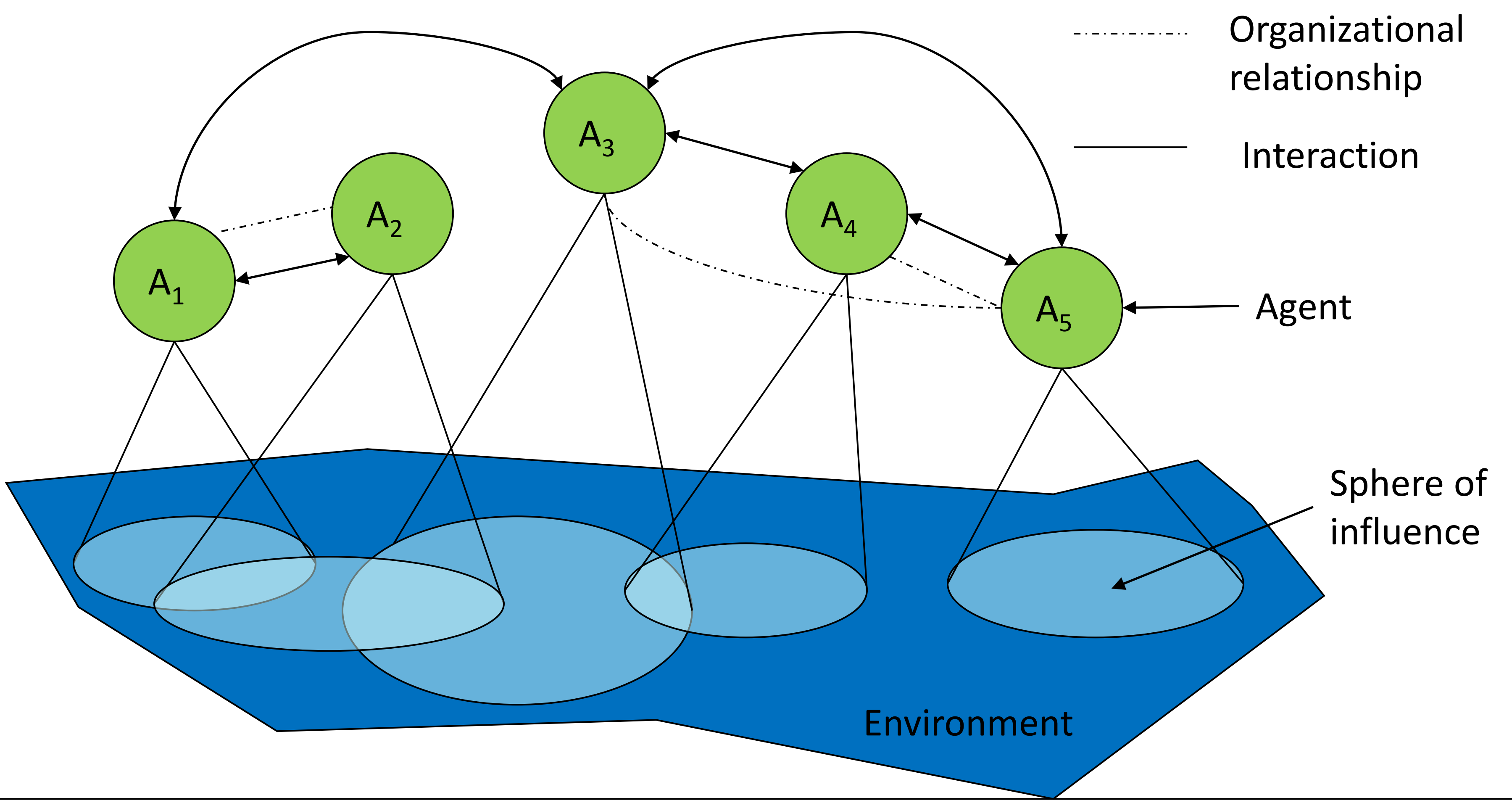
"A multi-agent system (MAS) can be defined as a loosely coupled network of problem solvers that interact to solve problems that are beyond the individual capabilities or knowledge of each problem solver."

*Durfee and Lesser, 1989*

## Characteristics of MAS

- The participants are self-interested.
- The participants and their capabilities changes over time, i.e. open systems.
- Each participant has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint.
- There is no system global control.
- Data is decentralized.
- Computation is asynchronous.

# View of a Canonical MAS



## Motivations for MAS

- Model human interactions.
- Solve very large and open problems.
- Interconnect and interoperate legacy system, information sources, or experts.
- Solve problems that naturally can be regarded as a society of agents.
- Enhance computational efficiency, reliability, extensibility, maintainability, flexibility, and reuse.

## Multiagent Systems is Interdisciplinary

- The field of Multiagent Systems is influenced and inspired by many other fields:
  - Economics
  - Philosophy
  - Game Theory
  - Logic
  - Ecology
  - Social Sciences
- What makes the multiagent systems field unique is that it emphasizes that the agents in question are *computational, information processing* entities.

## Agent Design, Society Design

- Agent design: How do we build agents capable of independent, autonomous action, so that they can successfully carry out tasks we delegate to them?
- Society design: How do we build agents that are capable of interacting (cooperating, coordinating, negotiating) with other agents in order to successfully carry out those delegated tasks, especially when the other agents cannot be assumed to share the same interests/goals?

## Social Ability

- Cooperation is *working together as a team to achieve a shared goal*.
  - Often prompted either by the fact that no one agent can achieve the goal alone, or that cooperation will obtain a better result (e.g., get result faster).
- Coordination is *managing the interdependencies between activities*.
- Negotiation is *the ability to reach agreements on matters of common interest*.
  - Typically involves offer and counter-offer, with compromises made by participants.

## Working Together

- Why and how should agents work together?
  - Since agents are autonomous, they have to make decisions at *run-time* and be capable of *dynamic coordination*.
- Cooperation is *working together as a team to achieve a shared goal*.
  - Often prompted either by the fact that no one agent can achieve the goal alone, or that cooperation will obtain a better result (e.g., get result faster).
- Coordination is *managing the interdependencies between activities*.
- Negotiation is *the ability to reach agreements on matters of common interest*.
  - Typically involves offer and counter-offer, with compromises made by participants.
- Overall they will need to be able to share:
  - Tasks
  - Information
- If agents are designed by different individuals, they may not have common goals.

## Criteria for Assessing Agent-based Systems

- *Coherence* - how well the [multiagent] system behaves as a unit along some dimension of evaluation (Bond and Gasser).
  - We can measure coherence in terms of solution quality, how efficiently resources are used, conceptual clarity and so on.
- *Coordination* - the degree… to which [the agents]… can avoid “extraneous” activity [such as]… synchronizing and aligning their activities (Bond and Gasser).
  - If the system is perfectly coordinated, agents will not get in each others’ way, in a physical or a metaphorical sense.

## Solution Approaches

- Sophisticated individual agents
- Organizations
- Task allocation and multiagent planning
- Recognizing and resolving conflicts
- Modeling other agents
- Communication
- Managing resources
- Adaptation and learning

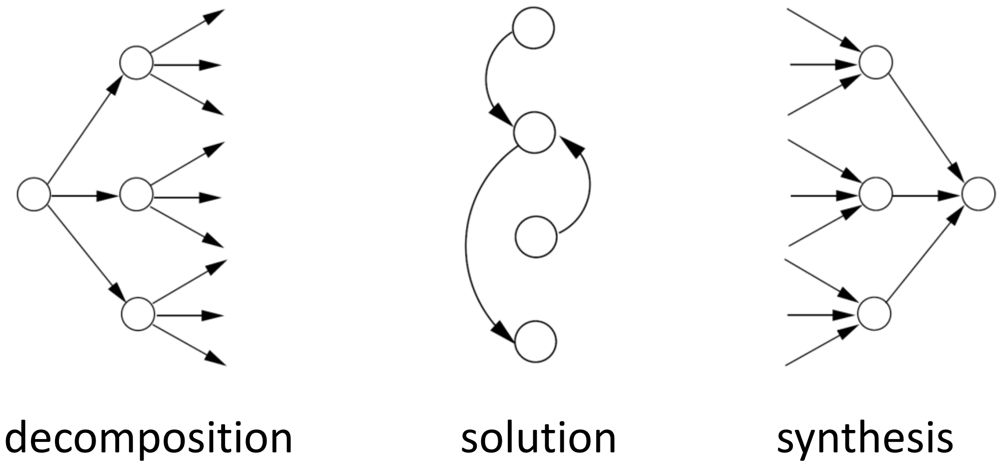
## Research Challenges in MAS

- How to formulate, describe, decompose, and allocate problems and synthesize results among a group of agents.
- How to enable agents to communicate and interact.
- How to make sure that agents act coherently in making decision or taking action.
- How to enable individual agents to represent and reason about the actions, plans, and knowledge of other agents to coordinate with them.
- How to recognize and reconcile disparate viewpoints and conflicting intentions among a collection of agents trying to coordinate their actions.
- How to engineer and constrain practical multi-agent systems.
- How to design technology platforms and development methodologies for MASs.

## Distributed Constraint Solving

# Cooperative Problem Solving

- How does a group of agents work together to solve problems?
- If we "own" the whole system, we can design agents to help each other whenever asked. In this case, we can assume agents are benevolent: our best interest is their best interest.
- Problem-solving in benevolent systems is *cooperative distributed problem solving* (CDPS).
- There are three stages:
  - Problem decomposition
  - Sub-problem solution
  - Answer synthesis



## Distributed Constraint Reasoning

- CSP: $(X, D, C)$
  - $X = \{x_1, x_2, \ldots, x_n\}$ *variables*
  - $D = \{d_1, d_2, \ldots, d_n\}$ *domains* (finite)
  - $C = \{c_1, c_2, \ldots, c_r\}$ *constraints*

  For each $c \in C$
  - $var(c) = \{x_i, x_j, \ldots, x_k\}$ *scope*
  - $rel(c) \in d_i \times d_j \times \ldots \times d_k$ *permitted tuples*
- Solution: total assignment satisfying all constraints
- DisCSP: $(X, D, C, A, \phi)$
  - $A = \{a_1, a_2, \ldots, a_k\}$ *agents*
  - $\phi: X \rightarrow A$ maps variables to agents
  - $c$ is known by agents owning $var(c)$

# Distributed Constraint Reasoning

Common assumptions:

- Agents communicate by *sending messages*
- An agent can send messages to others, *iff it knows their identifiers*
- The *delay* transmitting a message is finite but random
- For any pair of agents, *messages are delivered in the order they were sent*
- Agents *know the constraints in which they are involved*, but not the other constraints
- Each agent owns a *single variable (agents = variables)*
- Constraints are *binary* (2 variables involved)

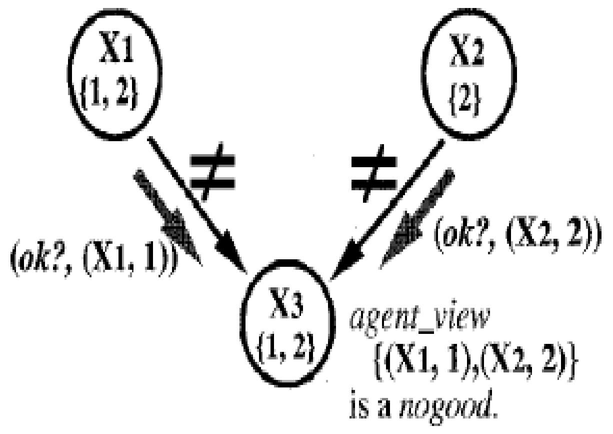
## Asynchronous Backtracking

- Each agent starts with instantiated variables, and knows all constraints that concern it
- Agent graph is connected, but not necessarily fully connected. Each agent has a set of values for the agents connected to it by incoming links (agent view)
- Agents can change their values or message agents that are linked to them
- Messages are either *Ok?* or *noGood*

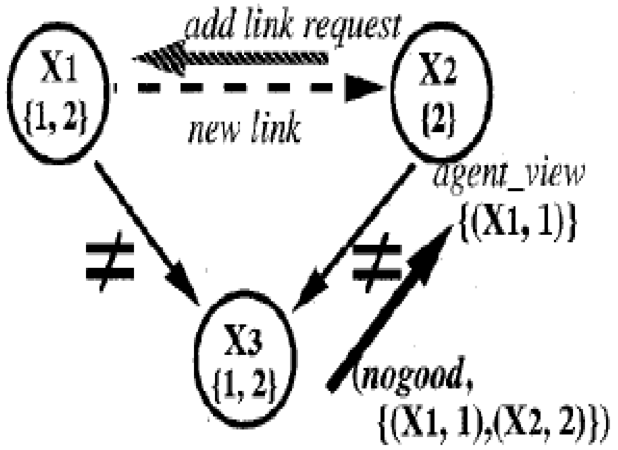
# Asynchronous Backtracking

- Agent view: the values of all agents linked to a particular agent
- Message Handling
  - *Ok?* -> Agent wants to know if it can assign a certain value to itself, so it asks another agent
    - Receiving agent updates agent view and checks for consistency, makes sure updated agent view is not a "noGood"
    - Oks only sent to lower priority agents
  - *NoGood* -> in evaluating an Ok? Message, an agent cannot find a value for itself that is consistent, then its updated agent view is noGood and a NoGood (backtracking) message is sent to another agent.
    - Nogoods only sent to higher priority agents
- NoGoods can be seen as derived constraints
- Preventing infinite loops by having a total order among agents for communication
  - Only need to know order of agents that one agent is linked to

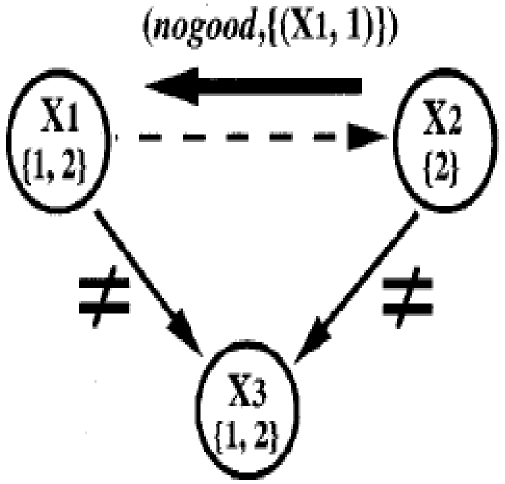
# Example: Asynchronous Backtracking



(a)



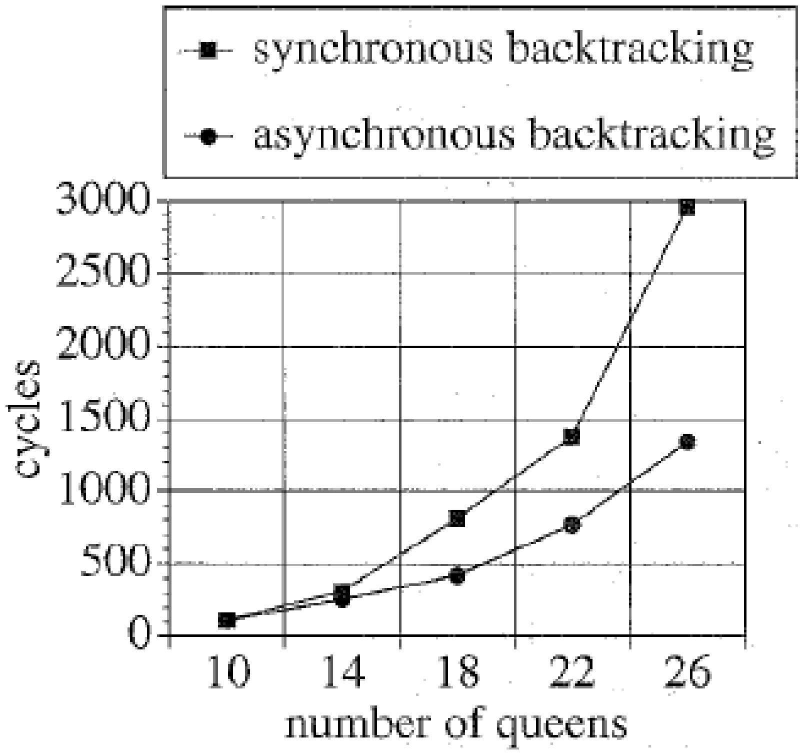
(b)



(c)

Source: M. Yokoo, E.H. Durfee, T. Ishida, K. Kuwabara, Distributed constraint satisfaction for formalizing distributed problem solving, in: 12th International Conference on Distributed Computing Systems (ICDCS-92), 1992, pp. 614–621.

## Comparison



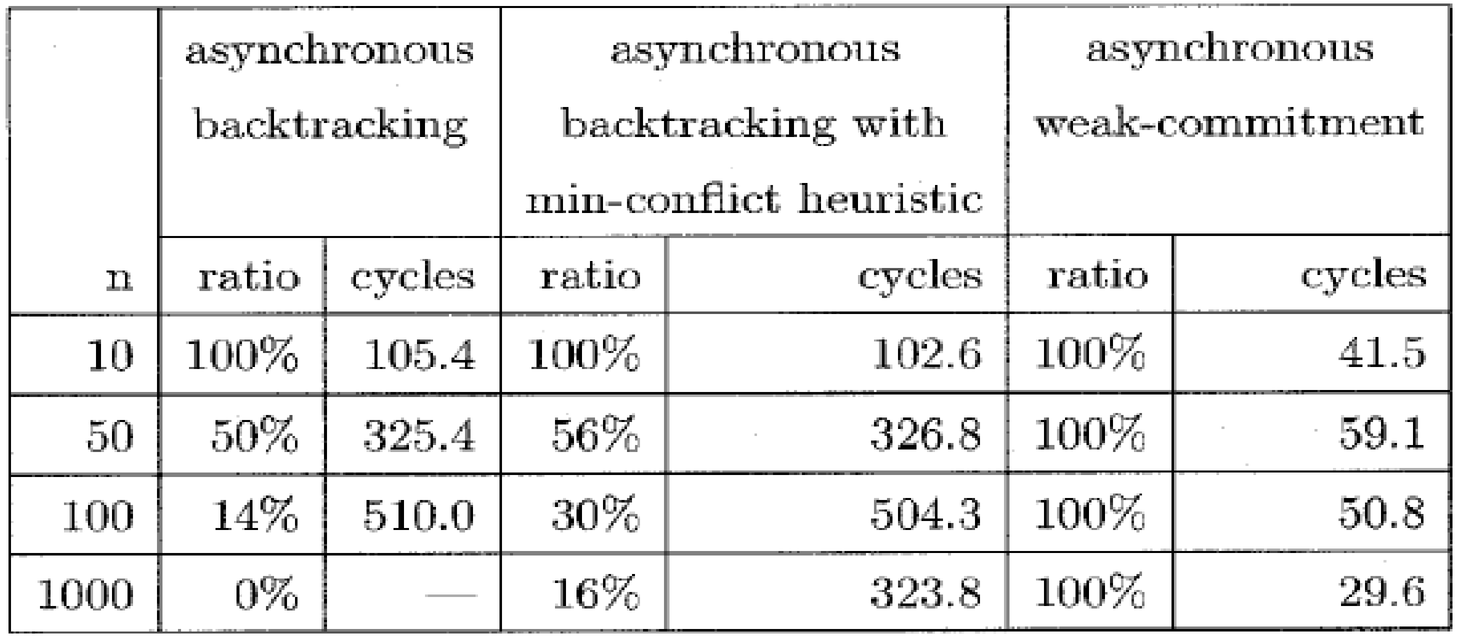
Source: M. Yokoo, E.H. Durfee, T. Ishida, K. Kuwabara, Distributed constraint satisfaction for formalizing distributed problem solving, in: 12th International Conference on Distributed Computing Systems (ICDCS-92), 1992, pp. 614–621.

## Asynchronous Weak-Commitment Search (AWC)

- Improvement over asynchronous backtracking
- Uses local dynamic priority values rather than static global ordering
- When an agent generates a nogood value, it promotes itself within its local network
- In ABT, an agent backtracks at dead-ends by sending a *nogood* to a higher priority agent
- in AWC, an agent gives up the attempt to satisfy its constraints and delegates the problem to other agents by raising its own priority

# Comparison

COMPARISON BETWEEN ASYNCHRONOUS BACKTRACKING AND ASYNCHRONOUS WEAK-COMMITMENT SEARCH (DISTRIBUTED N-QUEENS)



| | asynchronous backtracking | | asynchronous backtracking with min-conflict heuristic | | asynchronous weak-commitment | |
|---|---|---|---|---|---|---|
| n | ratio | cycles | ratio | cycles | ratio | cycles |
| 10 | 100% | 105.4 | 100% | 102.6 | 100% | 41.5 |
| 50 | 50% | 325.4 | 56% | 326.8 | 100% | 59.1 |
| 100 | 14% | 510.0 | 30% | 504.3 | 100% | 50.8 |
| 1000 | 0% | — | 16% | 323.8 | 100% | 29.6 |

Source: M. Yokoo, E.H. Durfee, T. Ishida, K. Kuwabara, Distributed constraint satisfaction for formalizing distributed problem solving, in: 12th International Conference on Distributed Computing Systems (ICDCS-92), 1992, pp. 614–621.

## Distributed Constraint Reasoning

- Exact algorithms (DisCSP and DCOP)
  - Asynchronous Backtracking (ABT),
  - Asynchronous Weak-Commitment Search (AWCS),
  - Asynchronous Distributed Optimization (ADOPT, BnB-ADOPT),
  - Distributed Pseudotree Optimization Procedure (DPOP)
- Approximate Algorithms with quality guarantees
  - k-optimality,
  - Bounded max-sum
- Approximate Algorithms without quality guarantees
  - Distributed Stochastic Algorithm (DSA),
  - Max-Sum

# Collaborative Unmanned Aircraft Systems

A principled approach to building collaborative intelligent autonomous systems for complex missions.

## Collaborative Unmanned Aircraft Systems

A principled approach to building collaborative intelligent autonomous systems for complex missions.

Challenges:

- Support humans and robots including legacy systems
- Support adjustable autonomy and mixed-initiative interaction
- Manage tasks and information on many abstraction levels
- Coordinating control, reaction and deliberation
- Coordination of systems, resources and platforms
- Incomplete information at design time and run time
- Inspection, monitoring, diagnosis and recovery on many abstraction levels

LINKÖPINGS UNIVERSITET

# Autonomous Systems at AIICS, Linköping University

Micro UAVs
weight < 500 g,
diameter < 50 cm

Yamaha RMAX
weight 95 kg,
length 3.6 m

LinkQuad weight ~1 kg, diameter ~70cm

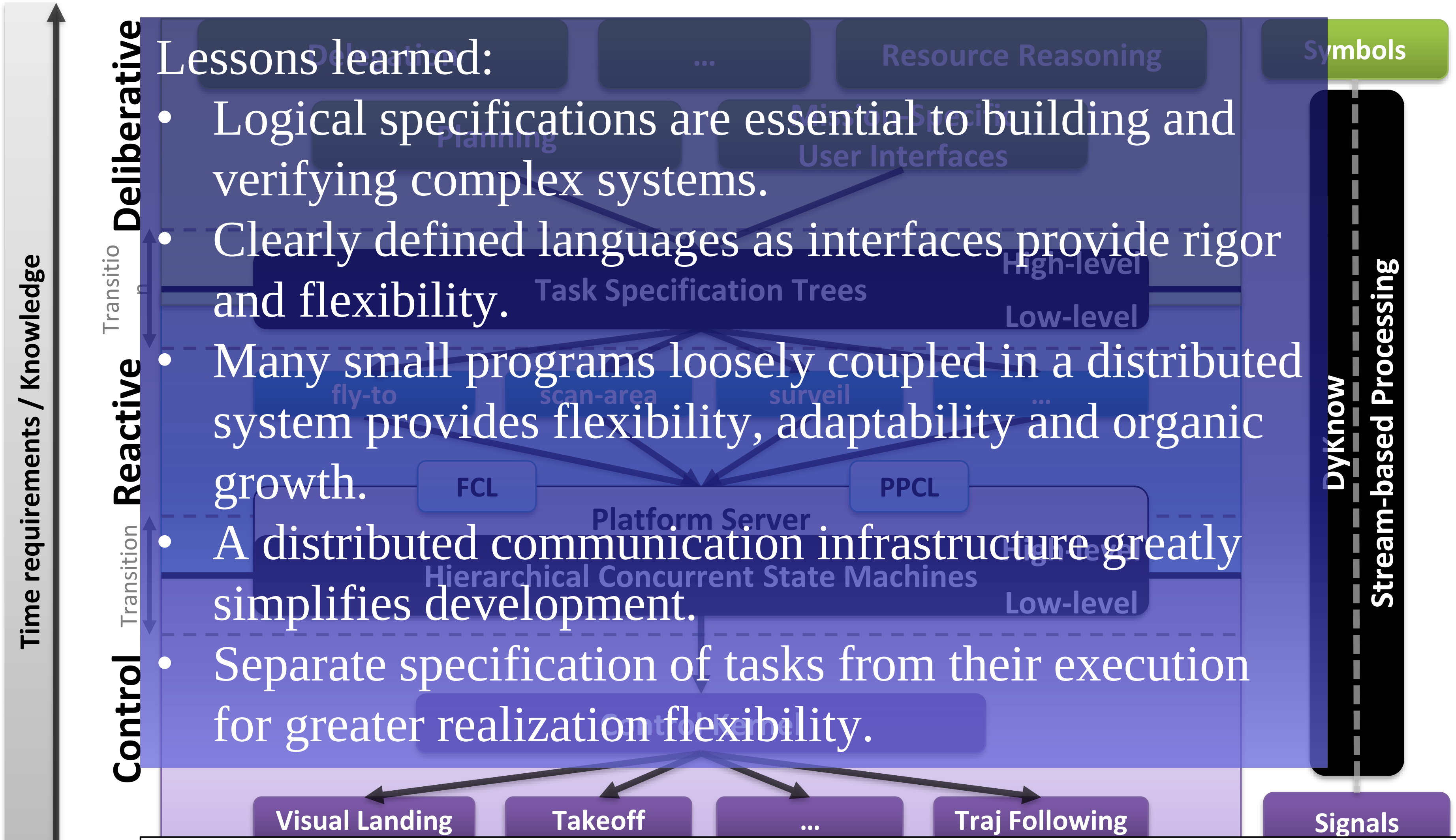
Selected Autonomous Functionalities

Linköping University, Sweden

Department of Computer and Information Science

Autonomous Unmanned Aerial Vehicle Technologies Lab

## HDRC3: A Distributed Hybrid Deliberative/Reactive Architecture for Autonomous Systems



Lessons learned:

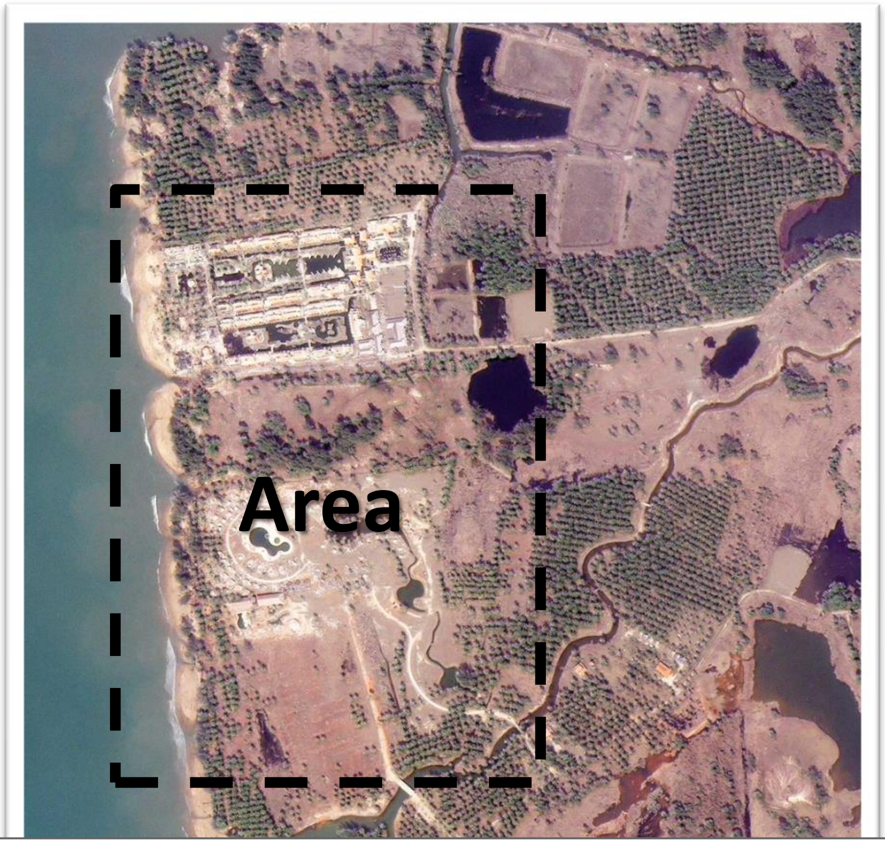
- Logical specifications are essential to building and verifying complex systems.
- Clearly defined languages as interfaces provide rigor and flexibility.
- Many small programs loosely coupled in a distributed system provides flexibility, adaptability and organic growth.
- A distributed communication infrastructure greatly simplifies development.
- Separate specification of tasks from their execution for greater realization flexibility.

P. Doherty, J. Kvarnström, M. Wzorek, P. Rudol, F. Heintz and G. Conte. 2014. **HDRC3 - A Distributed Hybrid Deliberative/Reactive Architecture for Unmanned Aircraft Systems**. In K. Valavanis, G. Vachtsevanos, editors, Handbook of Unmanned Aerial Vehicles, pages 849–952.

# Example Scenario: Search and Relief

Searching for injured people and delivering food, medicine and other supplies are highly prioritized activities in disaster relief.

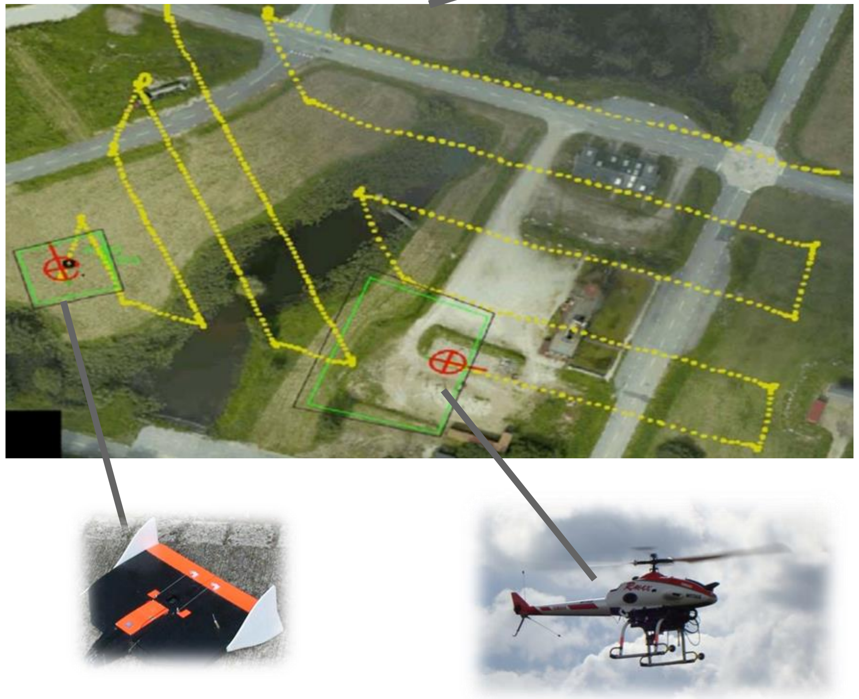
## Example Scenario: Search and Relief

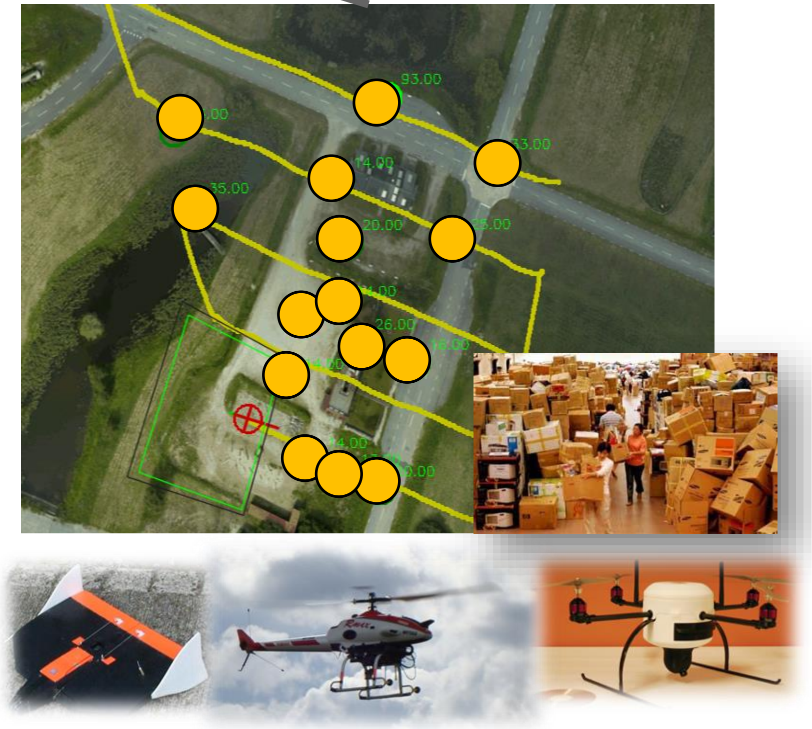


**Mission:** First scan Area for survivors, then deliver emergency supplies to the survivors.

# Example Scenario: Search and Relief

**Mission:** First scan Area for survivors, then deliver emergency supplies to the survivors.



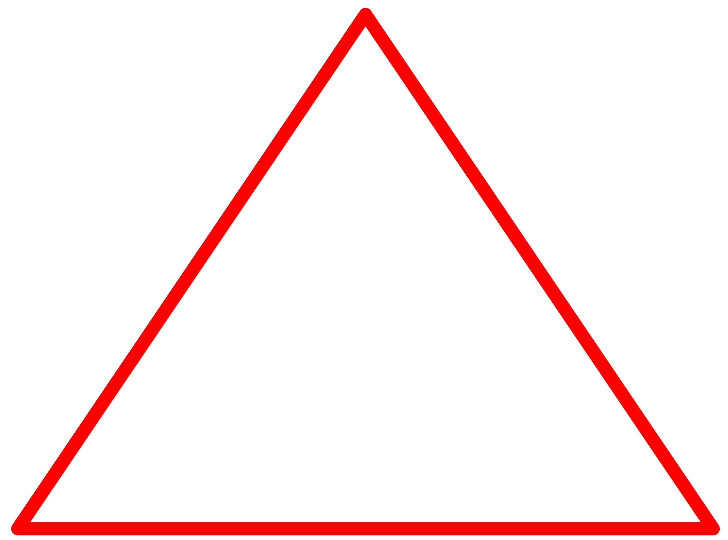


LINKÖPINGS UNIVERSITET

# Human-Robot Collaboration

Delegation

*Delegate(A, B, task, constraints)*



Adjustable Autonomy

By varying the task and constraints parameters the degree of autonomy allowed can be controlled.

Mixed-Initiative Interaction

*Delegate(GOP, UAV, task, constraints)*
*Delegate(UAV, GOP, task, constraints)*
Important: Safety, security, trust, etc.

LINKÖPINGS UNIVERSITET

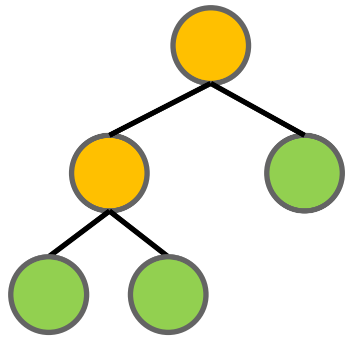
Patrick Doherty, Fredrik Heintz and Jonas Kvarnström. 2013.
High-level Mission Specification and Planning for Collaborative Unmanned Aircraft Systems using Delegation. Unmanned Systems, **1**(1):75–119. World Scientific.

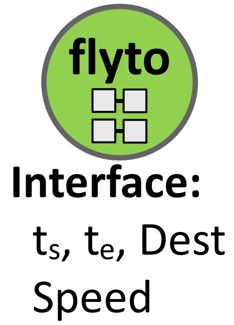
## Collaborative Tasks for UAS

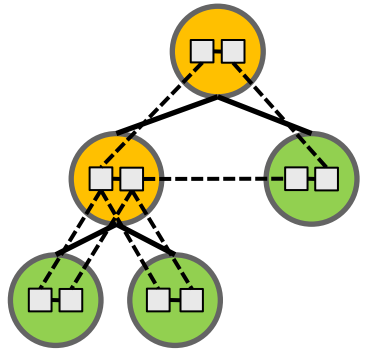
- Tasks need to be
  - general to cover the spectrum from high level goals to detailed plans (*task constraints*),
  - assigned to resource constrained physical platforms (*interrelated utilities*), and
  - expanded and modified as parts of tasks are recursively delegated (*complex tasks*).
- The task representation should
  - be highly flexible, distributed and dynamically extendible and
  - support dynamic adjustment of autonomy.

# Task Specification Trees

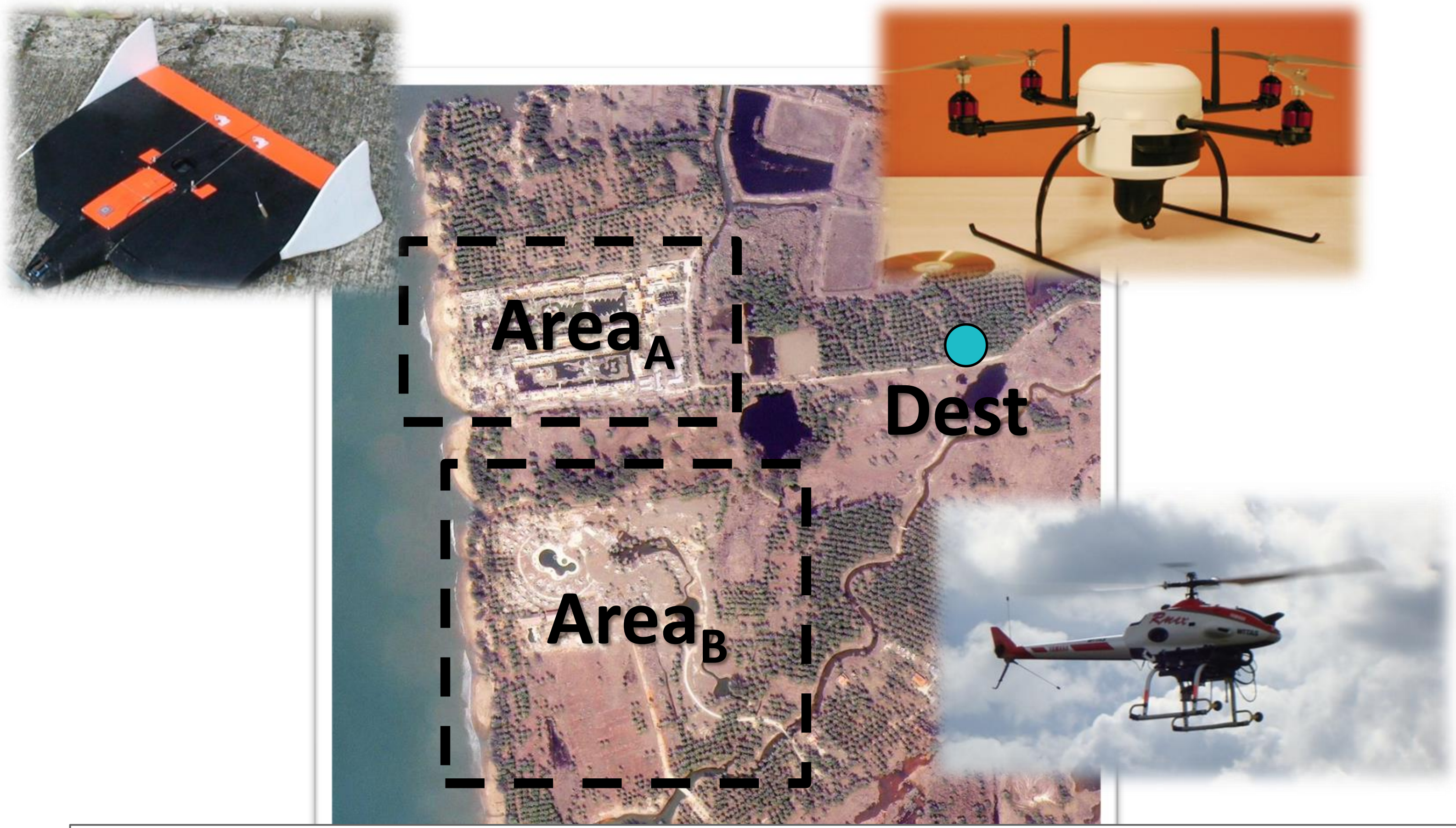
- A Task Specification Tree (TST) is a distributed data structure with a declarative representation that describes a complex multi-agent task.
- A node in a TST corresponds to a task. It has a node interface with parameters and a set of node constraints that restrict the parameters.
- There are currently six types of nodes: Sequence, concurrent, loop, select, goal, and elementary action.
- A TST is associated with a set of tree constraints expressing constraints between tasks in the tree.





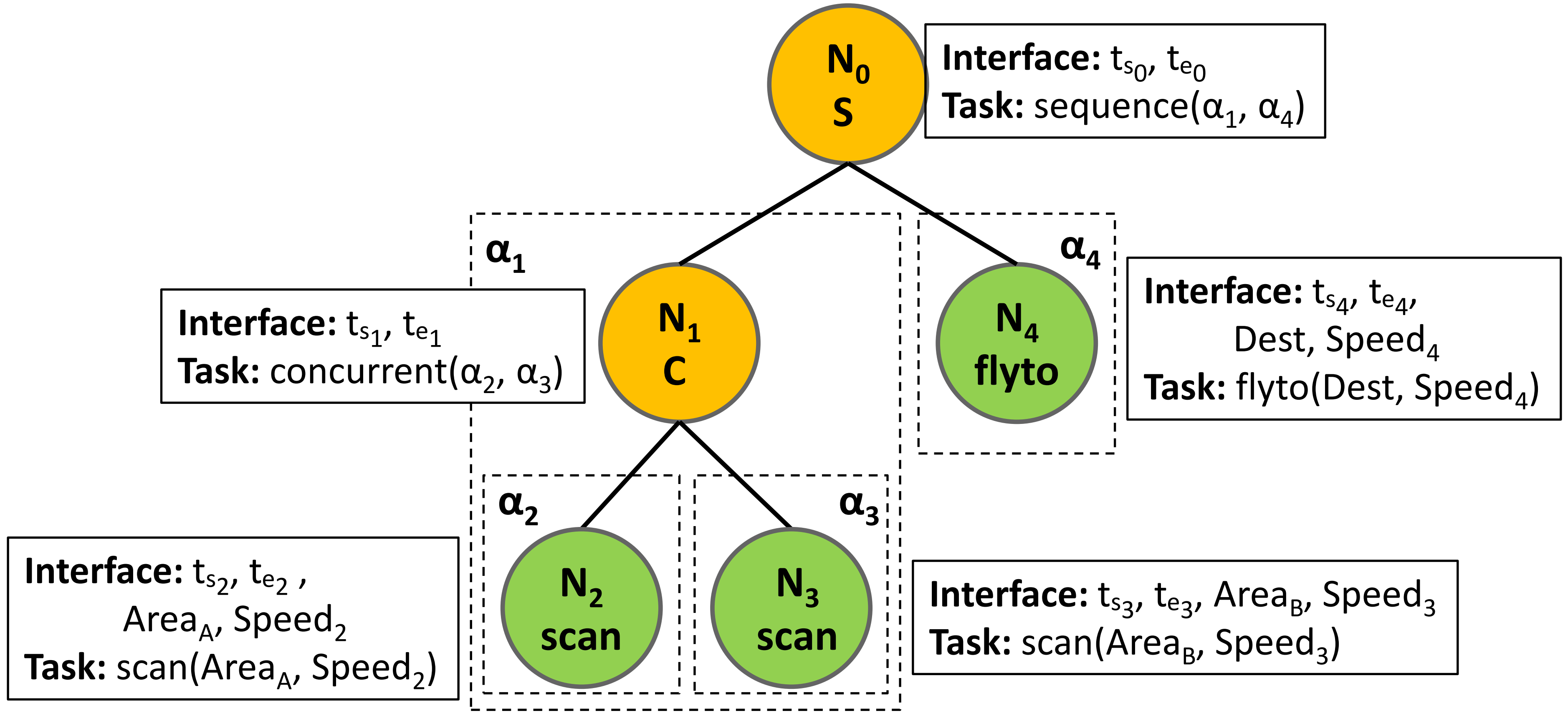


## Example Scenario: Search and Relief



**Mission:** First scan Area$_A$ and Area$_B$, then fly to Dest.

# Example TST

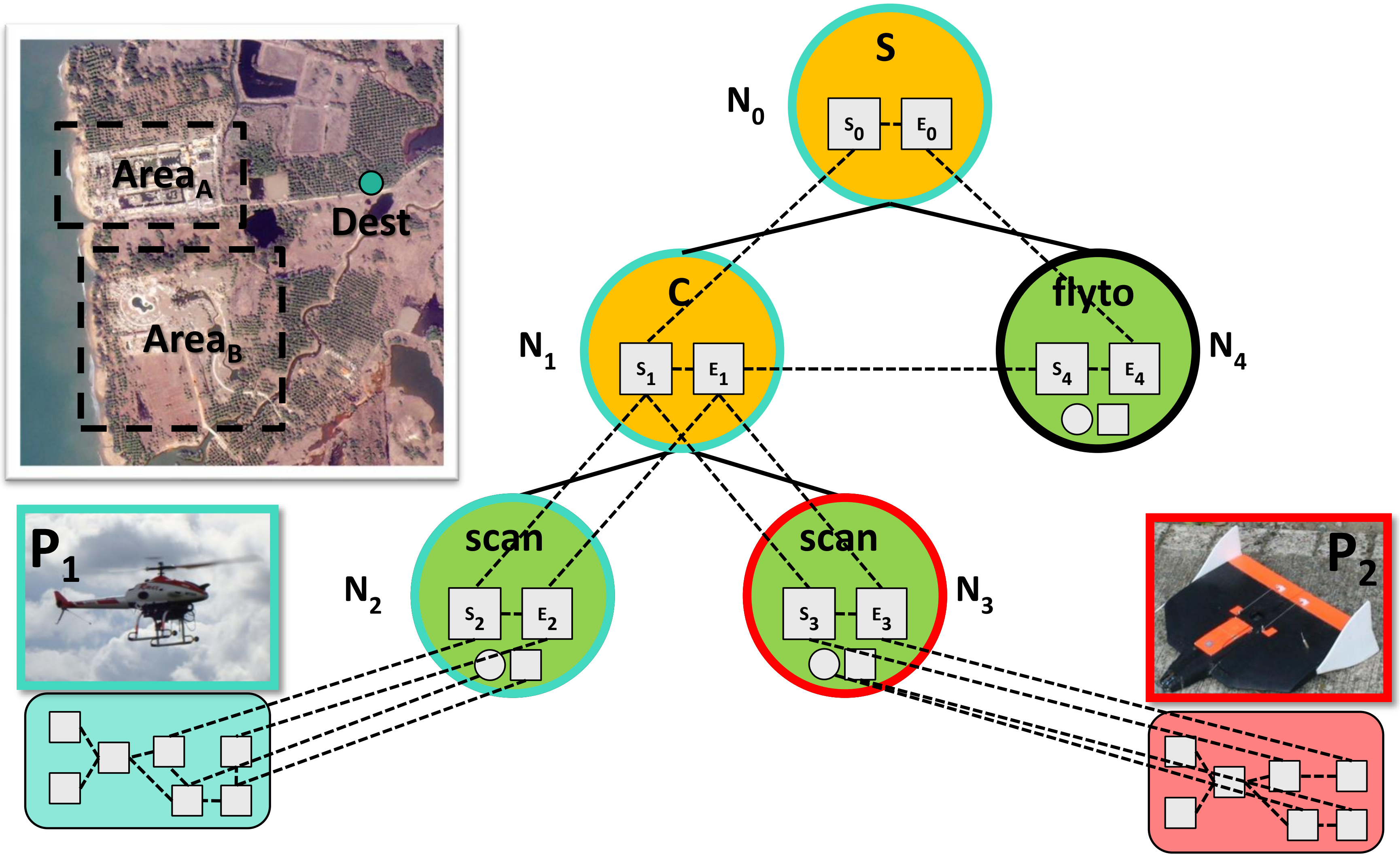

$N_0$
S
Interface: $t_{s_0}$, $t_{e_0}$
Task: sequence($\alpha_1$, $\alpha_4$)
$\alpha_1$
$N_1$
C
Interface: $t_{s_1}$, $t_{e_1}$
Task: concurrent($\alpha_2$, $\alpha_3$)
$\alpha_4$
$N_4$
flyto
Interface: $t_{s_4}$, $t_{e_4}$, Dest, $Speed_4$
Task: flyto(Dest, $Speed_4$)
$\alpha_2$
$N_2$
scan
Interface: $t_{s_2}$, $t_{e_2}$, $Area_A$, $Speed_2$
Task: scan($Area_A$, $Speed_2$)
$\alpha_3$
$N_3$
scan
Interface: $t_{s_3}$, $t_{e_3}$, $Area_B$, $Speed_3$
Task: scan($Area_B$, $Speed_3$)

**Mission:** First scan $Area_A$ and $Area_B$, then fly to Dest.

# Delegating TSTs

- What it means to be able to carry out a TST is defined in terms of the Can and Delegate predicates.
- Can($B$, $\tau$, [$t_s$, $t_{e, \dots}$], *cons*) asserts that an agent $B$ has the capabilities and resources for achieving a task $\tau$ in the interval [$t_s$, $t_e$] with the constraints *cons*.
- The semantics of control nodes is platform independent while the semantics of elementary action nodes are platform dependent.
  - Can($B$, $S(\tau_1, \dots, \tau_n)$, [$t_s$, $t_{e, \dots}$], *cons*) holds iff $B$ either can do or delegate each task $\tau_1, \dots, \tau_n$ in the sequence so that the constraints are satisfied.

# TST Delegation Example

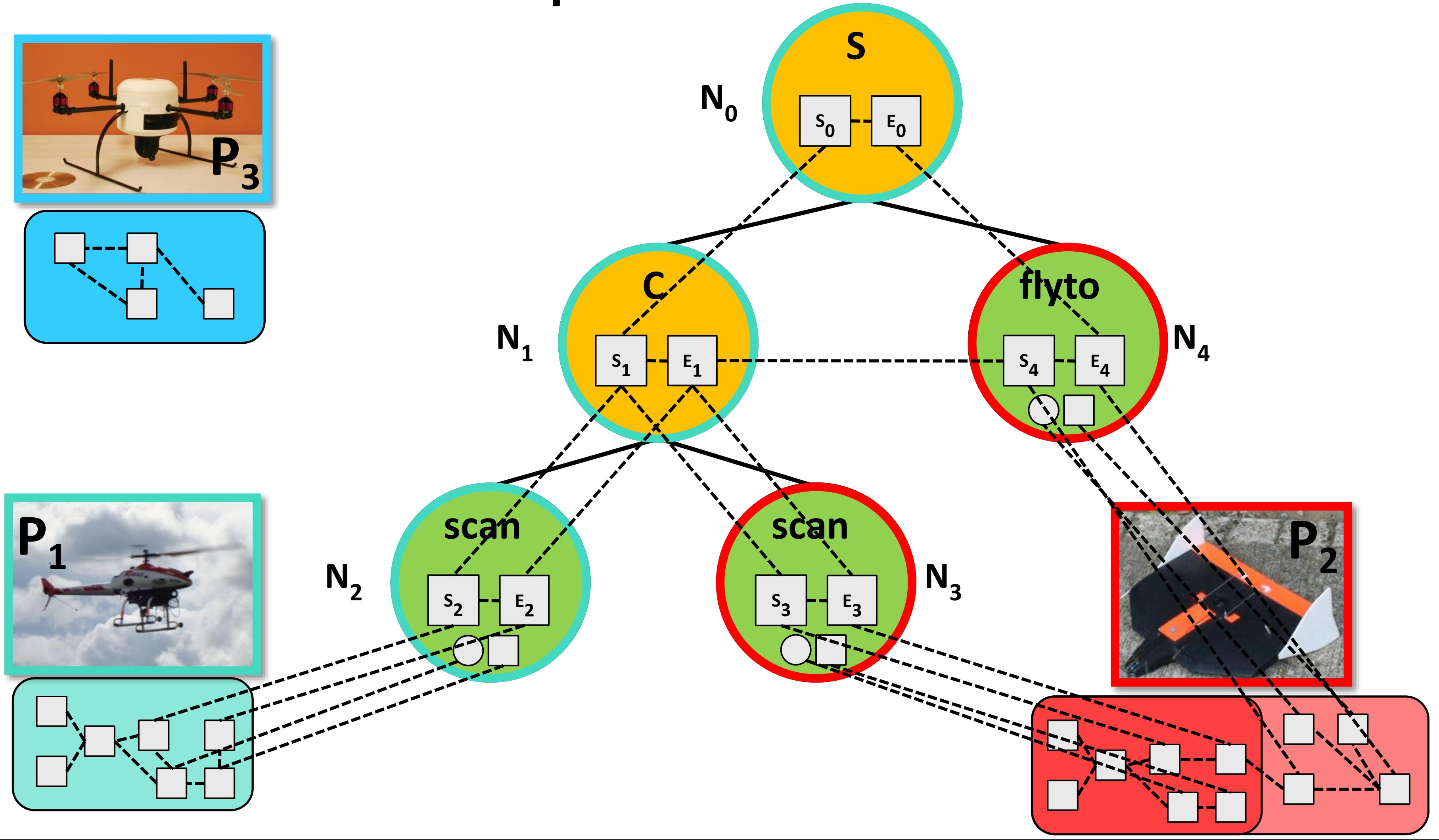


LINKÖPINGS UNIVERSITET

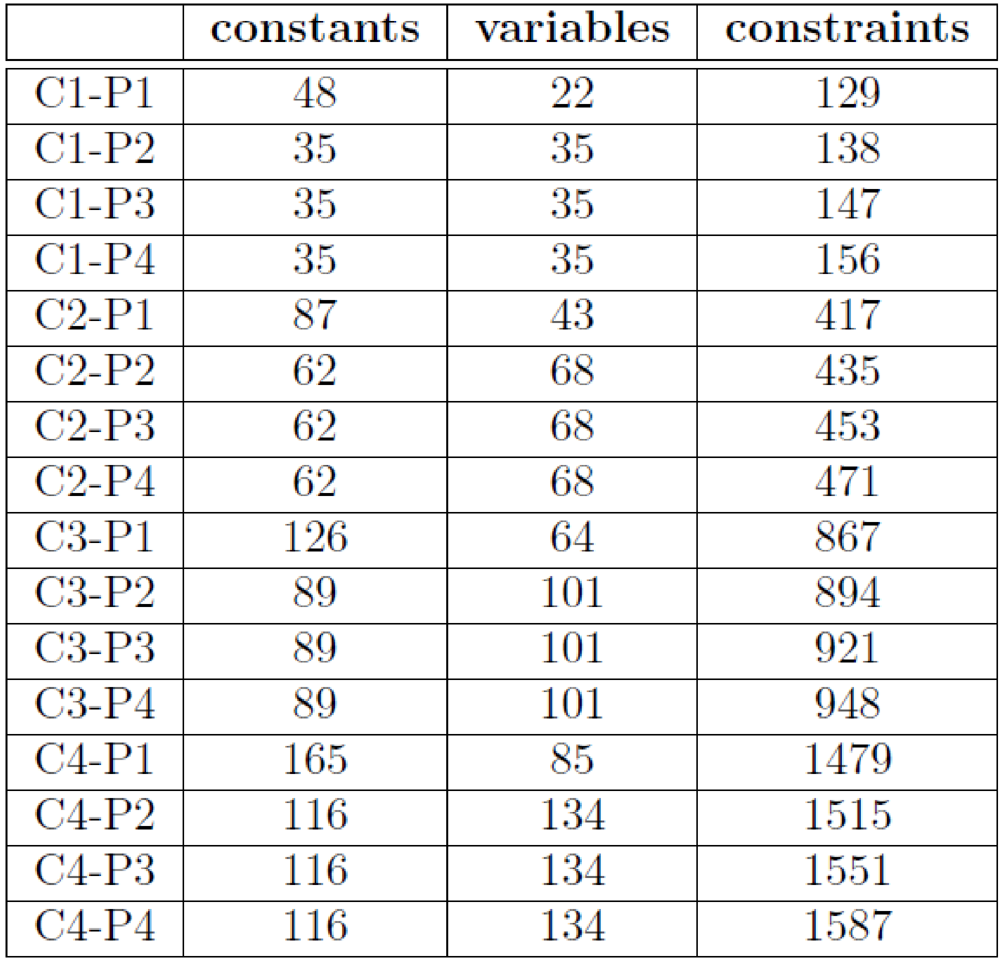
## Complex Task Allocation for CUAS

- The goal of the delegation process is to recursively find a set of platforms that can achieve a task specified as a TST.
- For a task to be achievable every node in the TST must be allocated to a platform such that the distributed constraint satisfaction problem corresponding to the semantics of the allocated TST is consistent.
- Hence we need to solve a complex task allocation problem.
- Our approach combines auction-based heuristic search for allocation and distributed constraint satisfaction for consistency checking partial allocations.

# TST Allocation Example

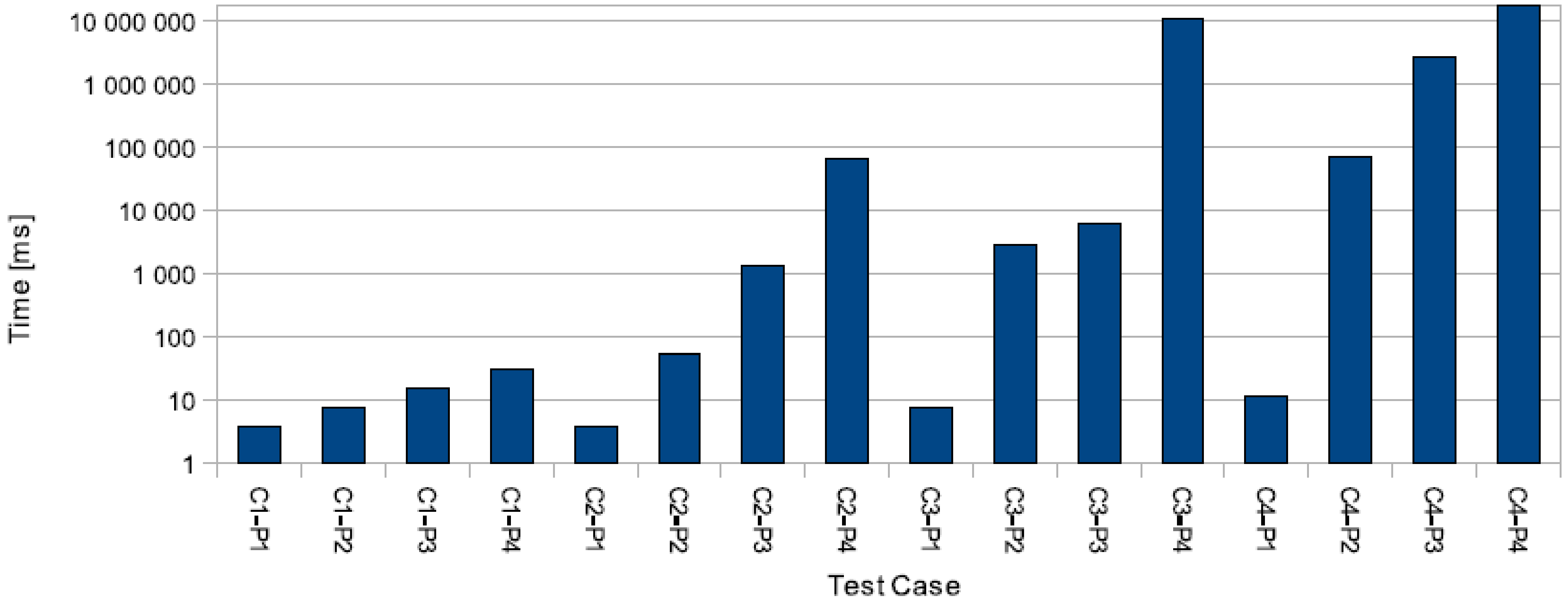


## Size of CSP Formulation

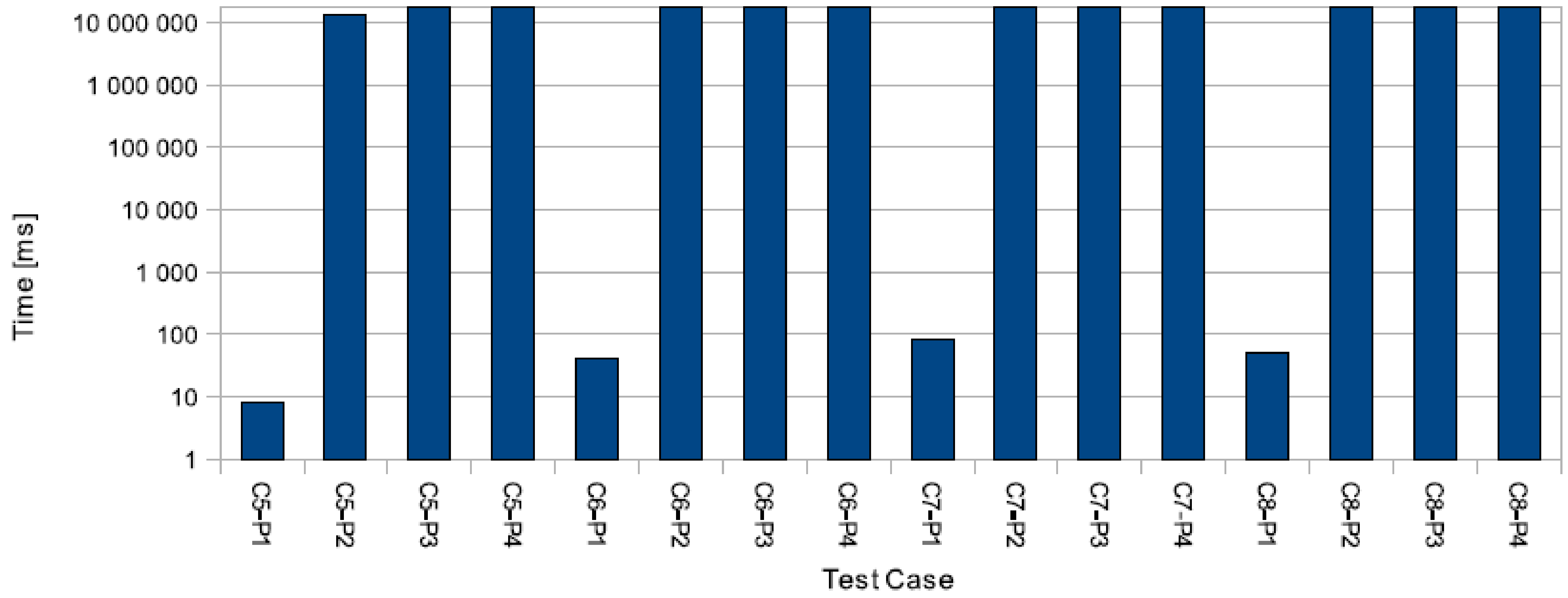


| | constants | variables | constraints |
|---|---|---|---|
| C1-P1 | 48 | 22 | 129 |
| C1-P2 | 35 | 35 | 138 |
| C1-P3 | 35 | 35 | 147 |
| C1-P4 | 35 | 35 | 156 |
| C2-P1 | 87 | 43 | 417 |
| C2-P2 | 62 | 68 | 435 |
| C2-P3 | 62 | 68 | 453 |
| C2-P4 | 62 | 68 | 471 |
| C3-P1 | 126 | 64 | 867 |
| C3-P2 | 89 | 101 | 894 |
| C3-P3 | 89 | 101 | 921 |
| C3-P4 | 89 | 101 | 948 |
| C4-P1 | 165 | 85 | 1479 |
| C4-P2 | 116 | 134 | 1515 |
| C4-P3 | 116 | 134 | 1551 |
| C4-P4 | 116 | 134 | 1587 |

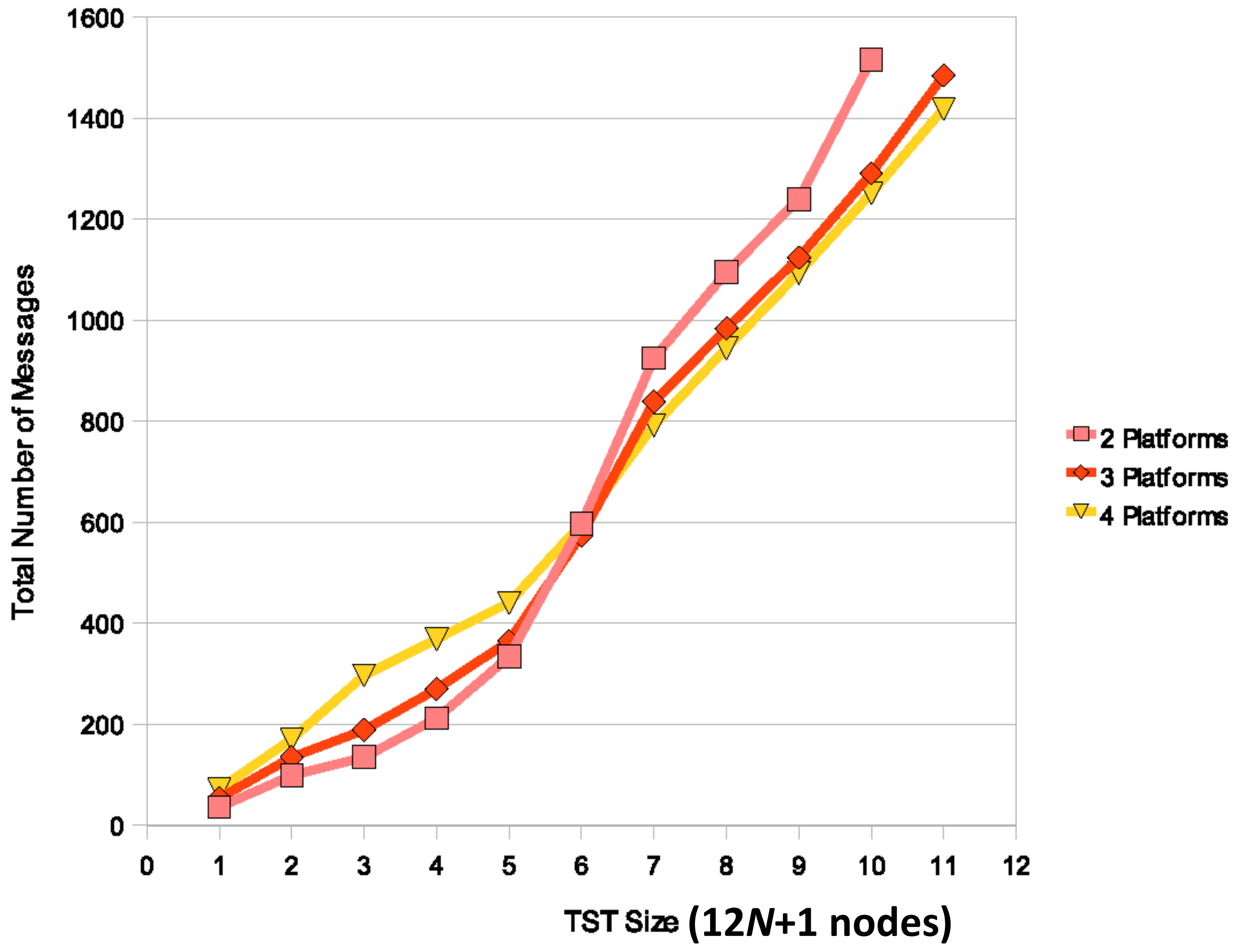
# Results Centralized CSP Formulation

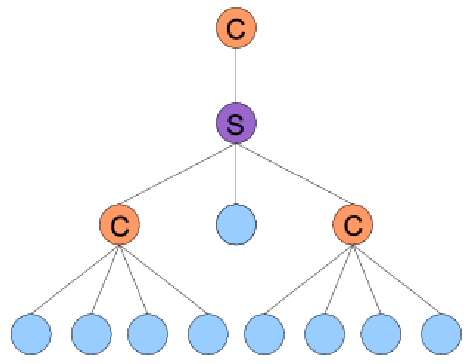


# Results Centralized CSP Formulation

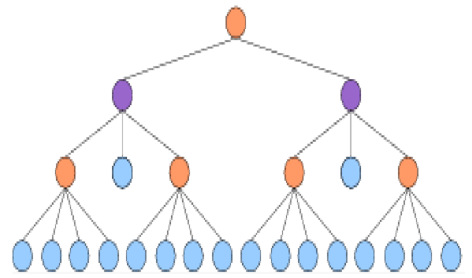


# Results Distributed CSP Formulation



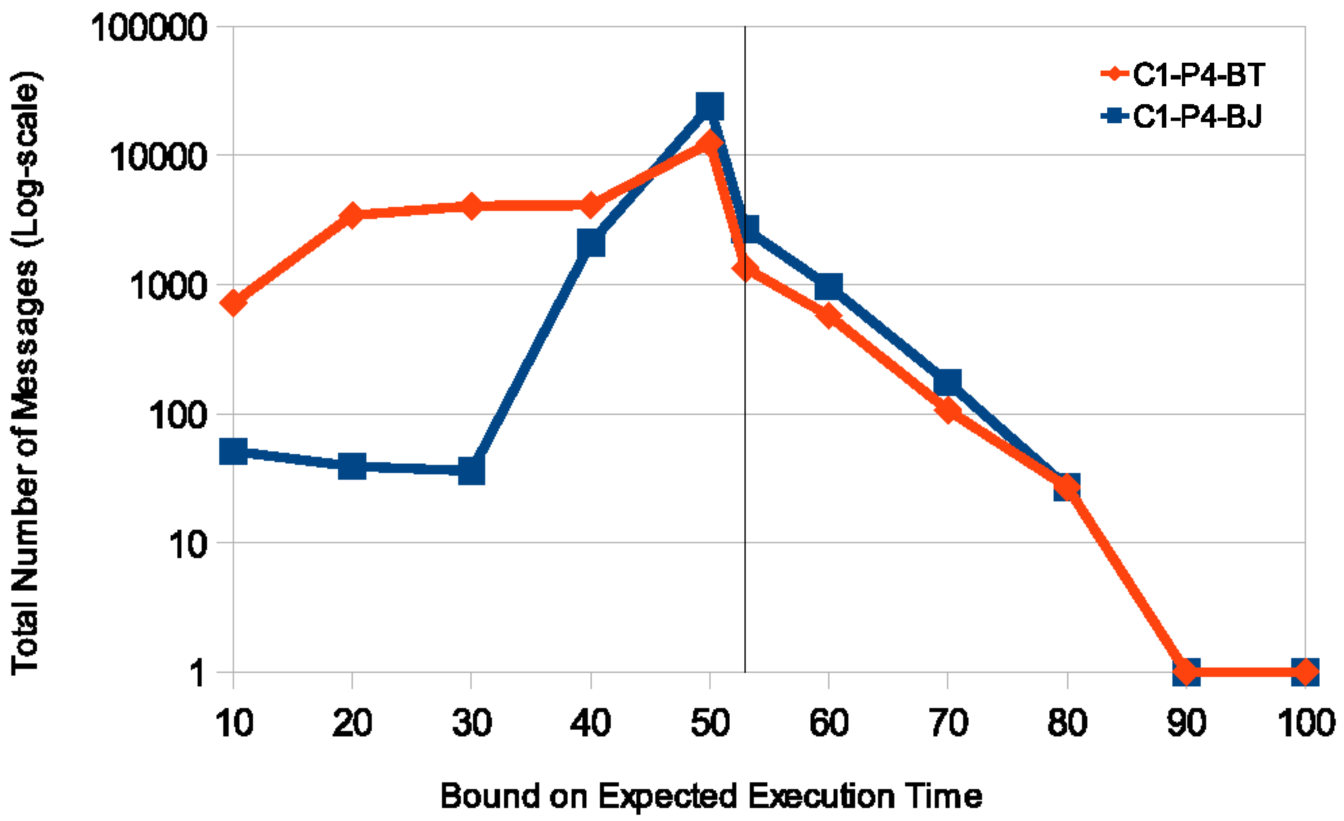


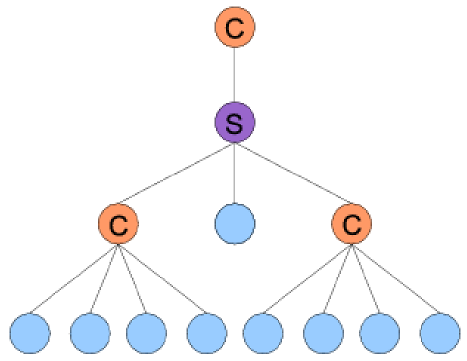
TST size 1 (13 nodes)



TST size 2 (25 nodes)

# Results Distributed CSP Formulation





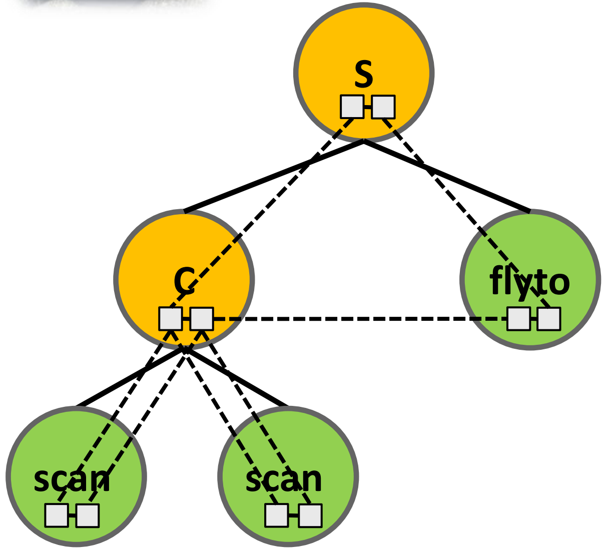
## Discussion

- Integrate planning and allocation
- Improve the efficiency of allocating TSTs
  - Study approximating algorithms for allocating TSTs
  - Study heuristics for allocating TSTs
  - Explicitly trade-off quality and efficiency (e.g. anytime algorithms)
  - Study restrictions on TSTs that facilitate more efficient allocation algorithms
  - Develop more efficient distributed constraint solving algorithms for our specific type of problems
  - Further study the interaction between auctions and constraint reasoning to balance guarantees and efficiency
- Consider optimization criteria such as
  - Maximize robustness to deviations due to uncertainty
  - Minimize total execution time and resources usage
  - Minimize resource usage and maximize communication quality

# Summary

- Discussed complex task allocation for collaborative unmanned aircraft systems.
- Outlined a delegation-based collaboration framework which uses Task Specification Trees (TSTs) for specifying complex tasks.
  - The consistency of allocations of platforms to TST nodes can be checked using distributed constraint satisfaction techniques.
  - To delegate a TST a complex task allocation problem has to be solved for example using a market-based approach.
- The result is a very rich collaborative robotics framework which opens up for many interesting research questions.

$Area_A$
Dest
$Area_B$



S
C
flyto
scan
scan