From Space to Smart Homes: Constraint-Based Planning for Domestic Assistance

Federico Pecora[⋆]

Center for Applied Autonomous Sensor Systems

Örebro University, SWEDEN

federico.pecora@oru.se

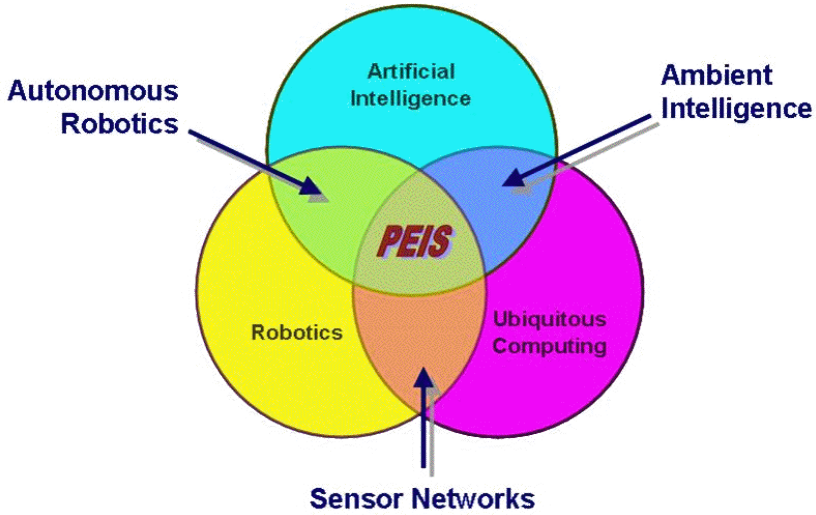
[⋆]Joint work w/ M. Cirillo, F. Dell'Osa, A. Loutfi, S. Magrelli, A. Saffiotti, J. Ullberg, F. Venturini

Outline

Outline

Ecologies of PEIS

- The PEIS-Ecology approach combines insights from several fields
- Grounded on an "ecological" vision of robotics [Saffiotti et al., 2008]
- Each individual in the ecology is a Physically Embedded Intelligent System



The PEIS Home

- The PEIS-Home is a prototypical smart home environment developed at AASS
- A number of PEIS have been realized: refrigerator, vision sensors, mobile robots, artificial noses, RFID-tagged floor and objects, ...

Synthesizing Intelligent Services in the PEIS-Home

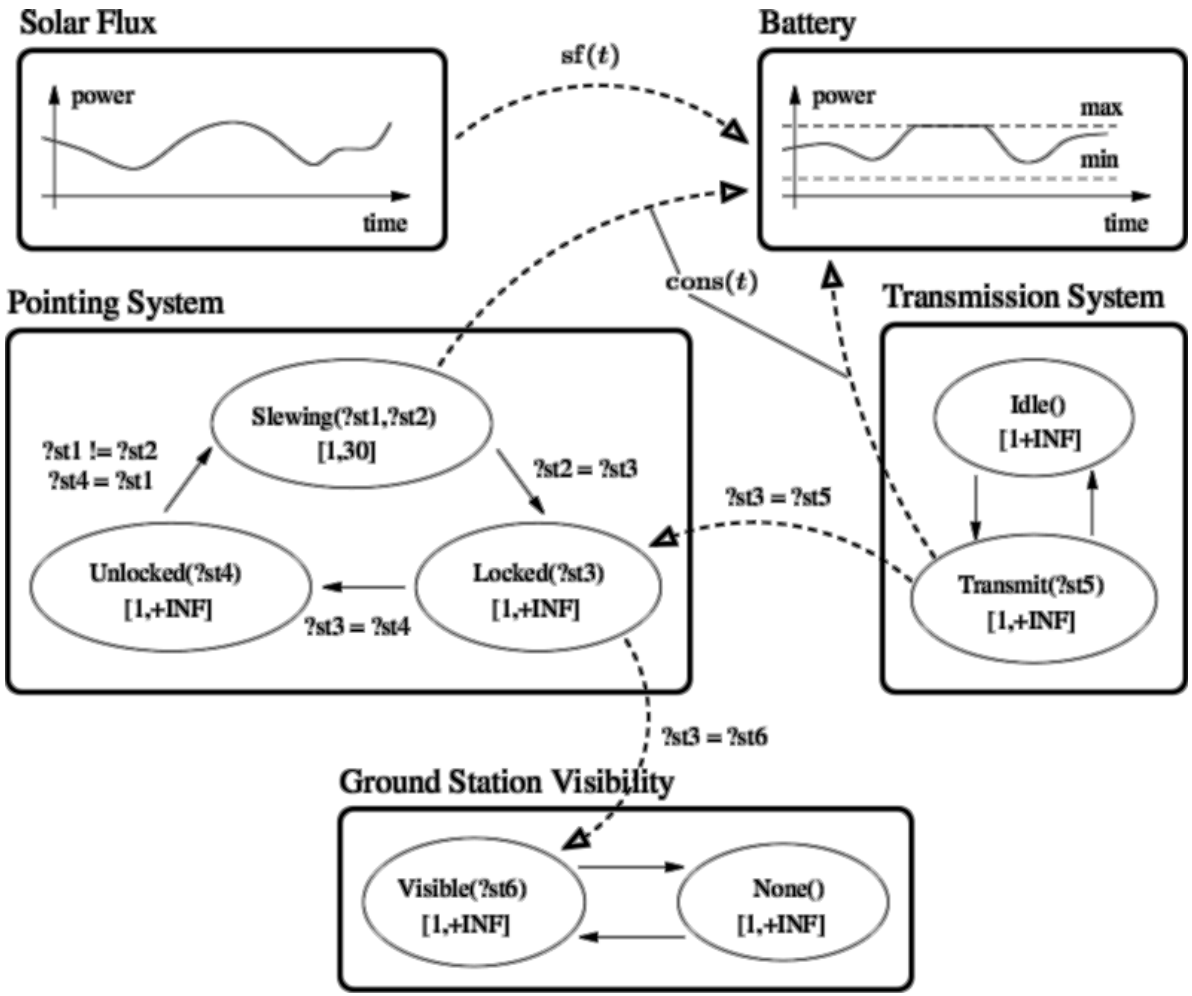
- **Activity recognition:** the ability of the intelligent system to deduce temporally contextualized knowledge regarding the state of the user
  - based on heterogeneous sensor readings and previously inferred knowledge
- **Planning and Execution:** the ability to proactively plan and execute services that provide contextualized assistance
  - based on the results of activity recognition
- This requires a way to model the temporal and causal dependencies that exist between sensor readings, tasks to be planned/executed and the state of the human user

Outline

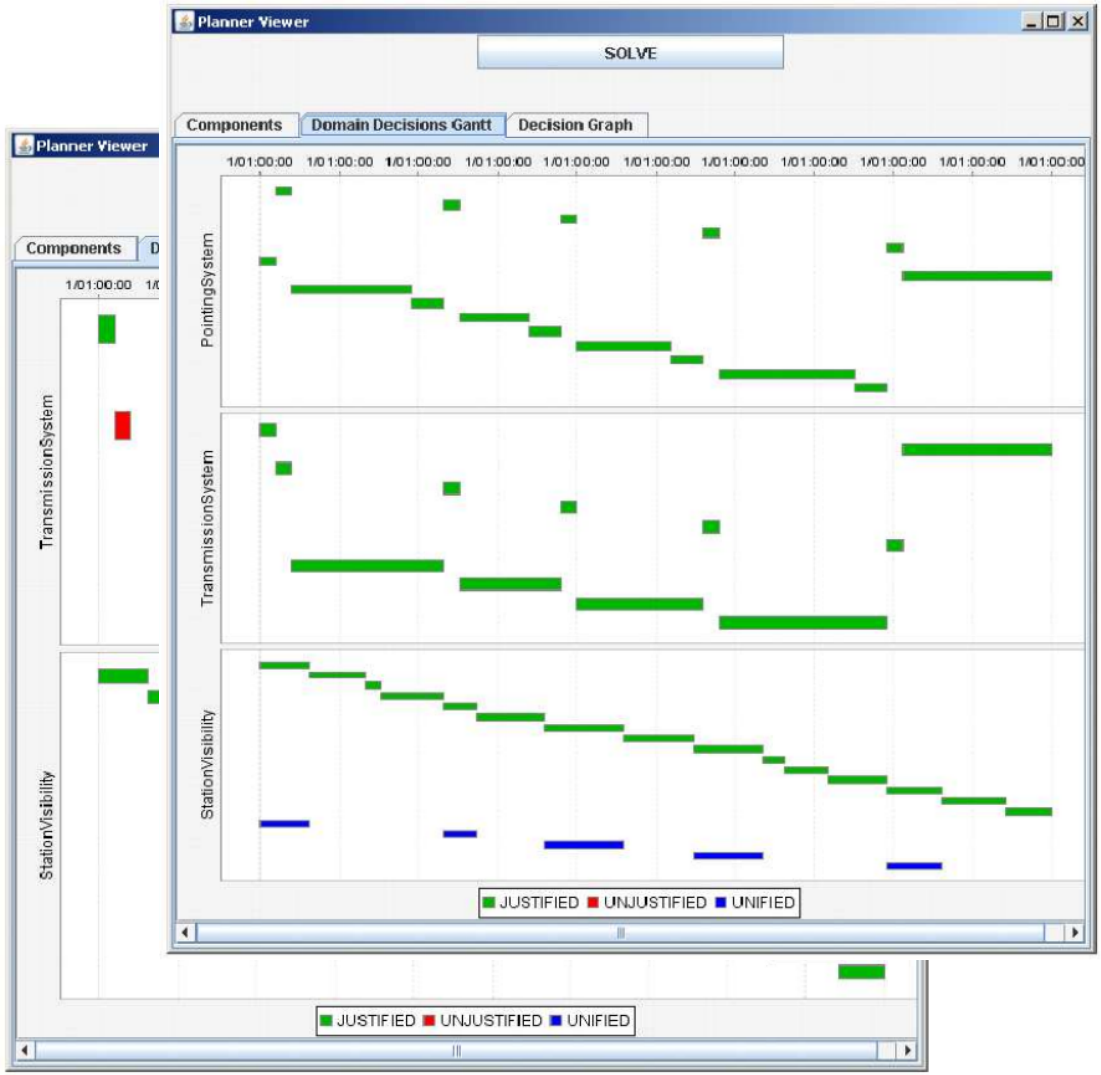
Constraint Reasoning for Domestic Plan Management

- The **SAM Activity Management architecture**: a constraint-based approach for activity recognition, planning and execution in PEIS Ecologies
- Based on the OMPS framework for constraint-based temporal reasoning [Fratini et al., 2008]
  - developed for ESA to improve the cost-effectiveness and flexibility of mission planning support tool development
  - used for Science Operations planning in the Mars Express mission [Cesta et al., 2008] and other domains [Cesta and Fratini, 2008]
- Grounded on the notions of component, decision, and constriant

Space Mission Planning in OMPS



Space Mission Planning in OMPS

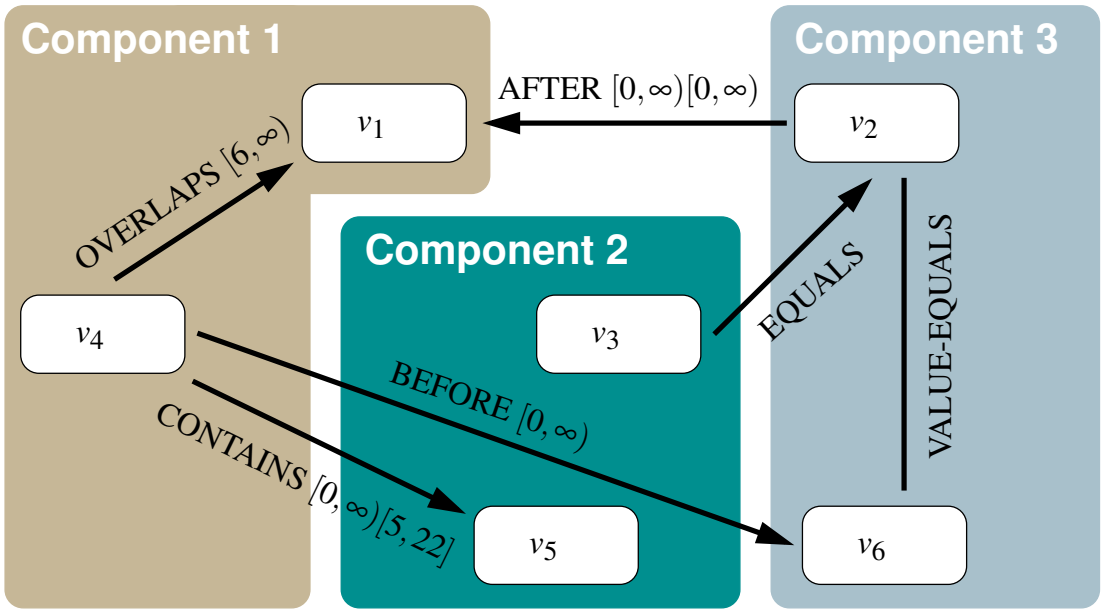


Constraint Reasoning in OMPS

- **Components** are logical or physical entities whose behavior evolves in time
  - functions of values in time, values can be numeric, symbolic, ...
  - e.g., **RoboticArm :** $\{\mathbf{grasp}, \mathbf{free}, \mathbf{drop}\}$
- **Decisions** are assertions on the value of a component in a flexible time interval
  - $d = \langle v, [I_s, I_e] \rangle$, where I_s, I_e are intervals of admissibility of start and end times of the decision
  - e.g., $\langle \mathbf{grasp}, [[5,5],[23,\infty)] \rangle$, $\langle \mathbf{grasp} \vee \mathbf{free}, [[2,7],[13,45]] \rangle$
- **Constraints** among the values or time intervals of decisions
  - temporal constraints extend the relations in Allen's interval algebra [Allen, 1984] and can involve decisions across components
  - e.g., $\langle \mathbf{drop}, [I_s, I_e] \rangle$ **DURING** $[0,\infty)[5,8]$ $\langle \mathbf{docked}, [I_s, I_e] \rangle$

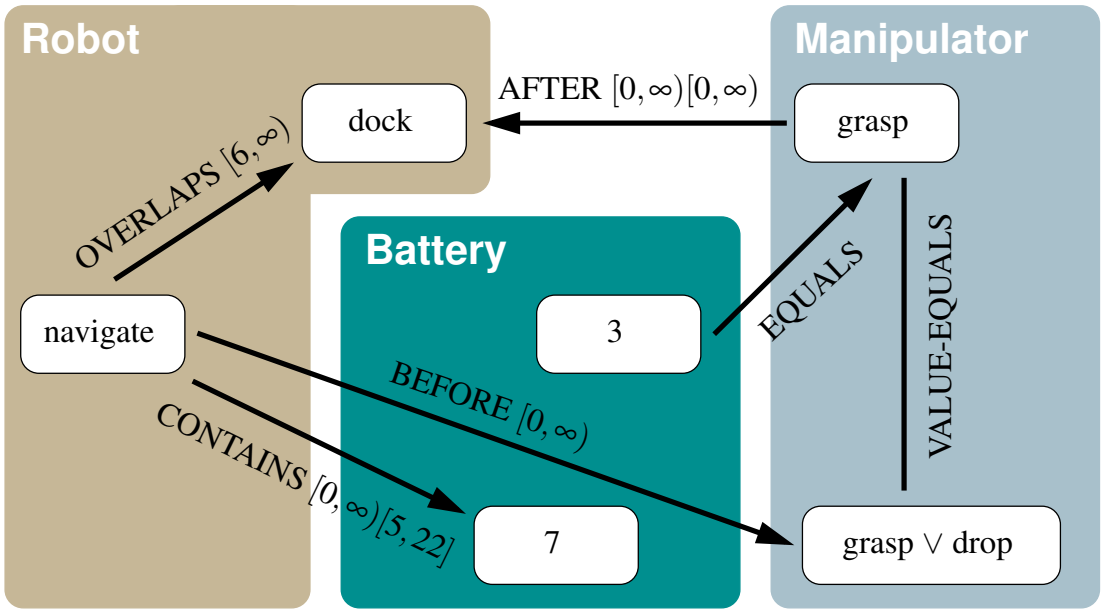
Constraint Reasoning in OMPS

- Decisions and constraints are maintained in a **Decision Network**
- Values of decisions depend on component type: consumable/renewable resources, state variables, . . .



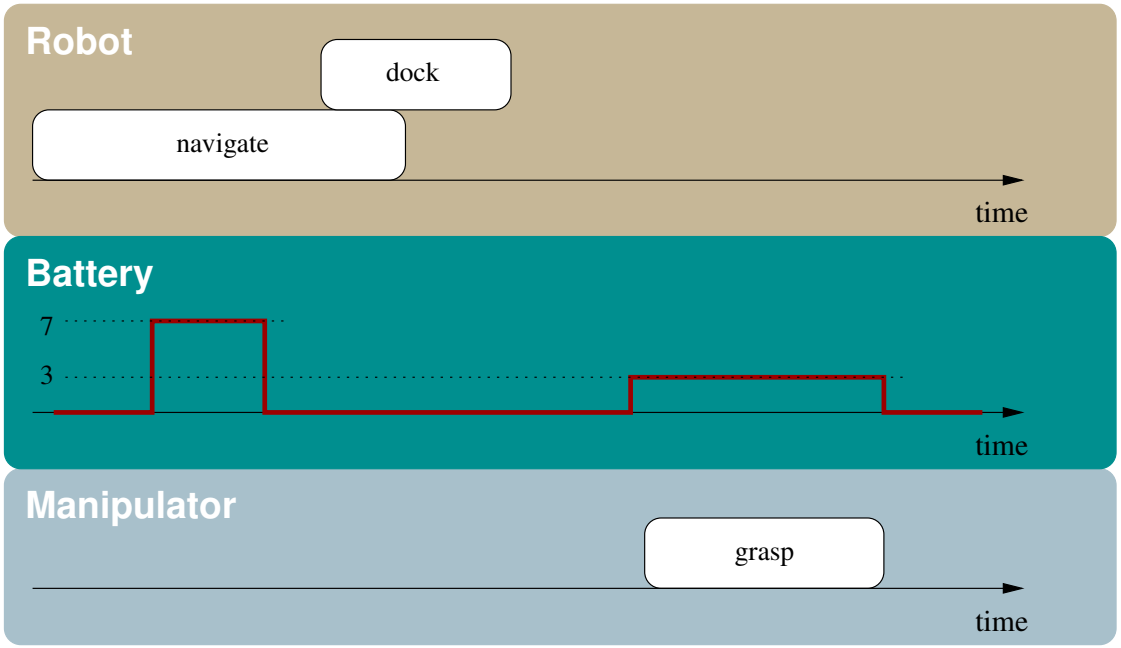
Constraint Reasoning in OMPS

- Decisions and constraints are maintained in a **Decision Network**
- Values of decisions depend on component type: consumable/renewable resources, state variables, ...

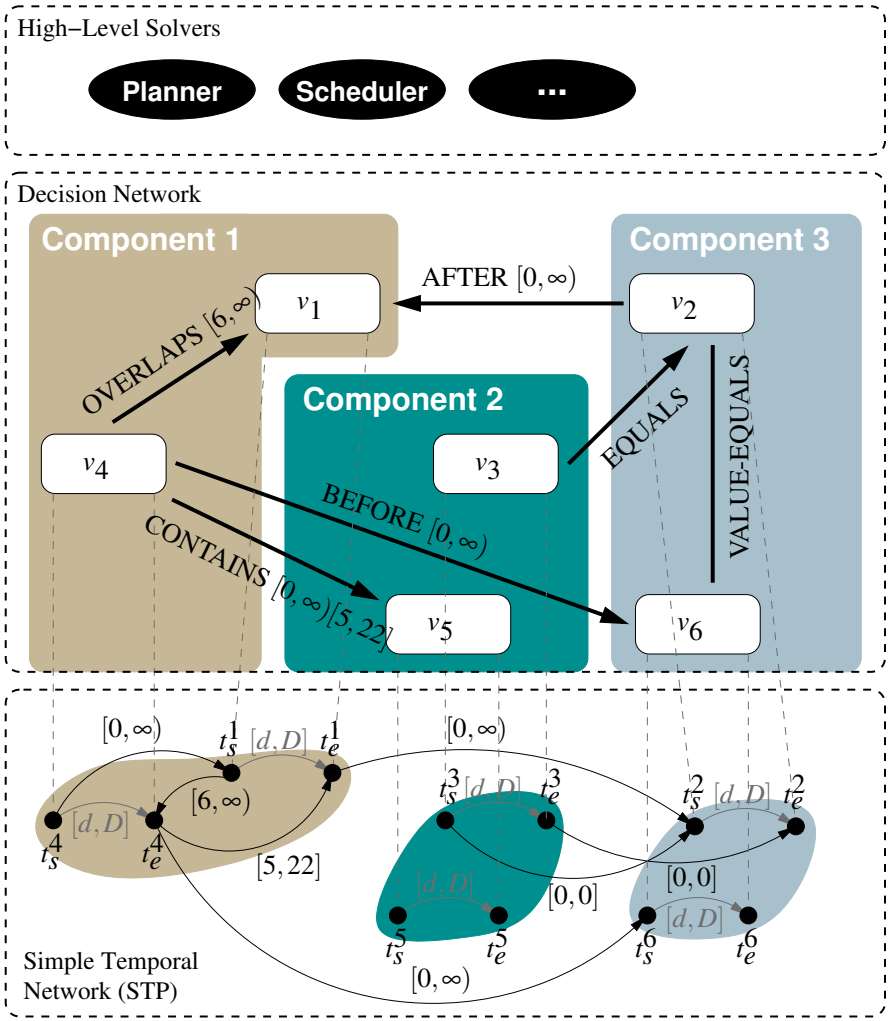


Constraint Reasoning in OMPS

- **Timeline:** the evolution in time of a component given the decision network
- Given the flexible nature of the decisions' temporal intervals, many timelines can be extracted (e.g., earliest start time)

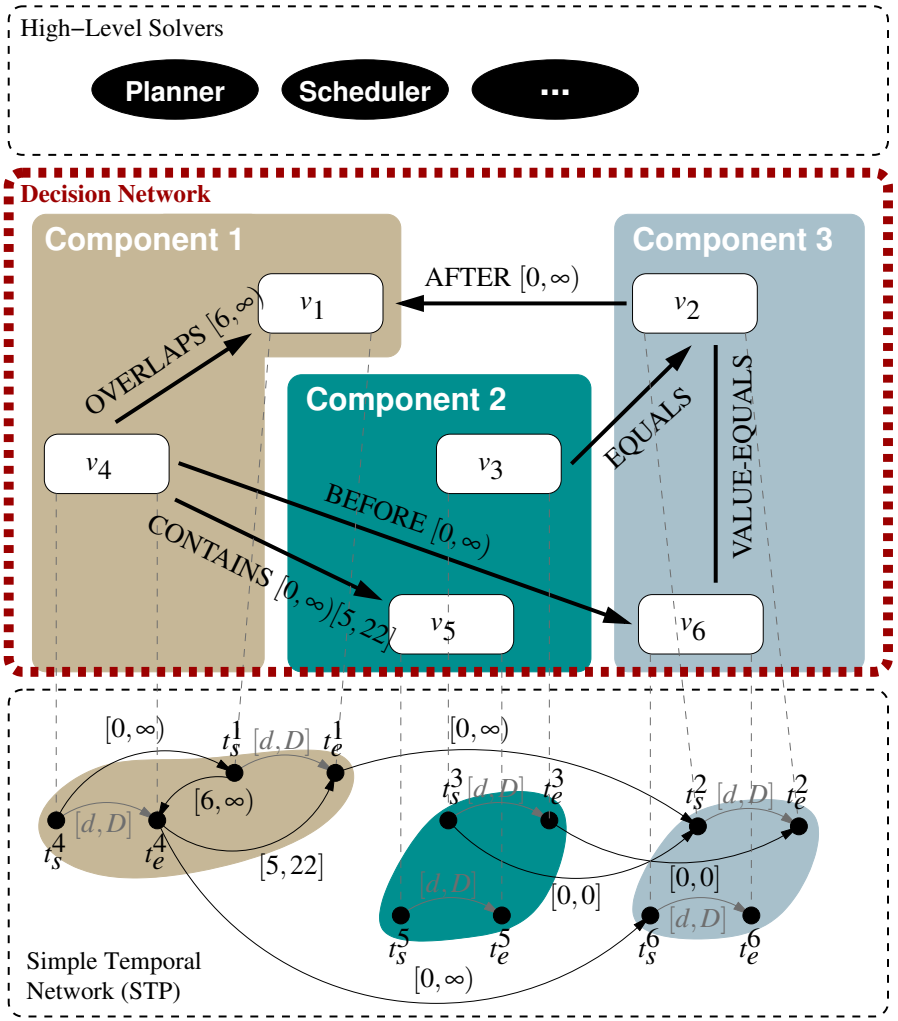


Constraint Reasoning in OMPS



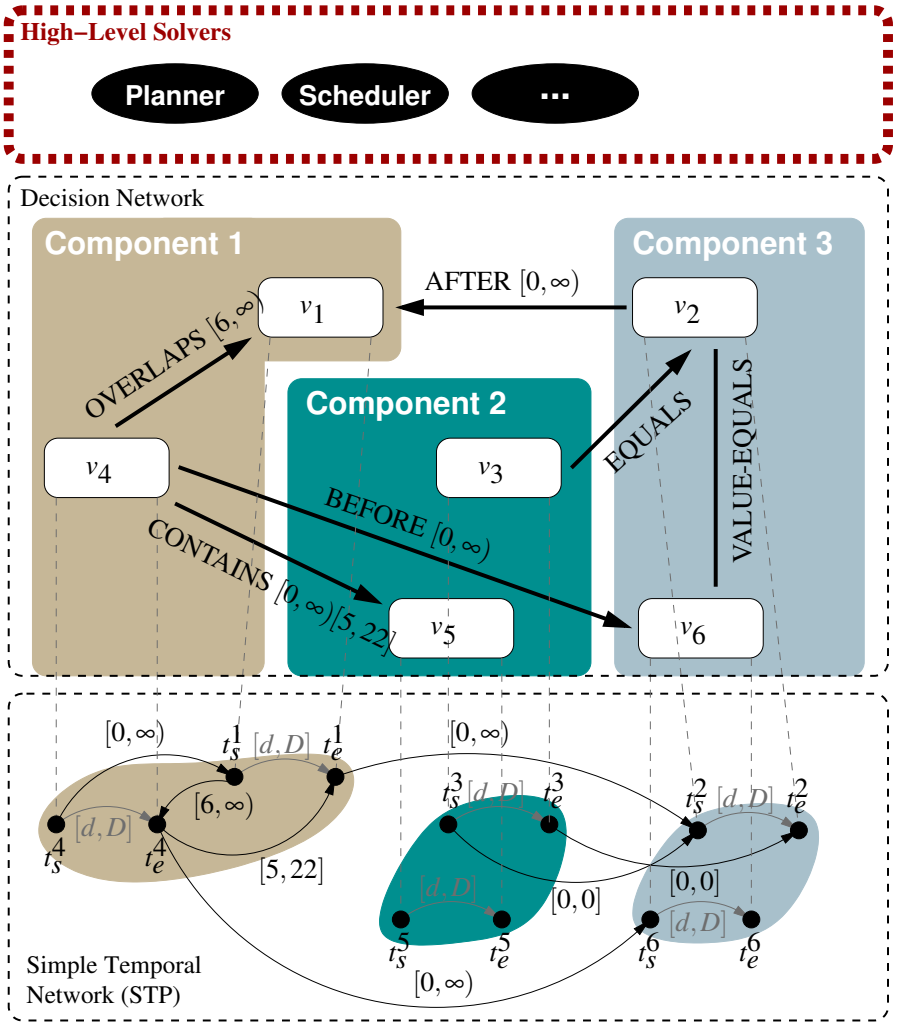
- OMPS **is a three-tiered architecture in which the Decision Network allows to share information between tiers**
- DN represents the current state of computation
- High-level solvers add/remove decisions and constraints to the DN
- OMPS provides a framework to implement new high-level reasoning modules

Constraint Reasoning in OMPS



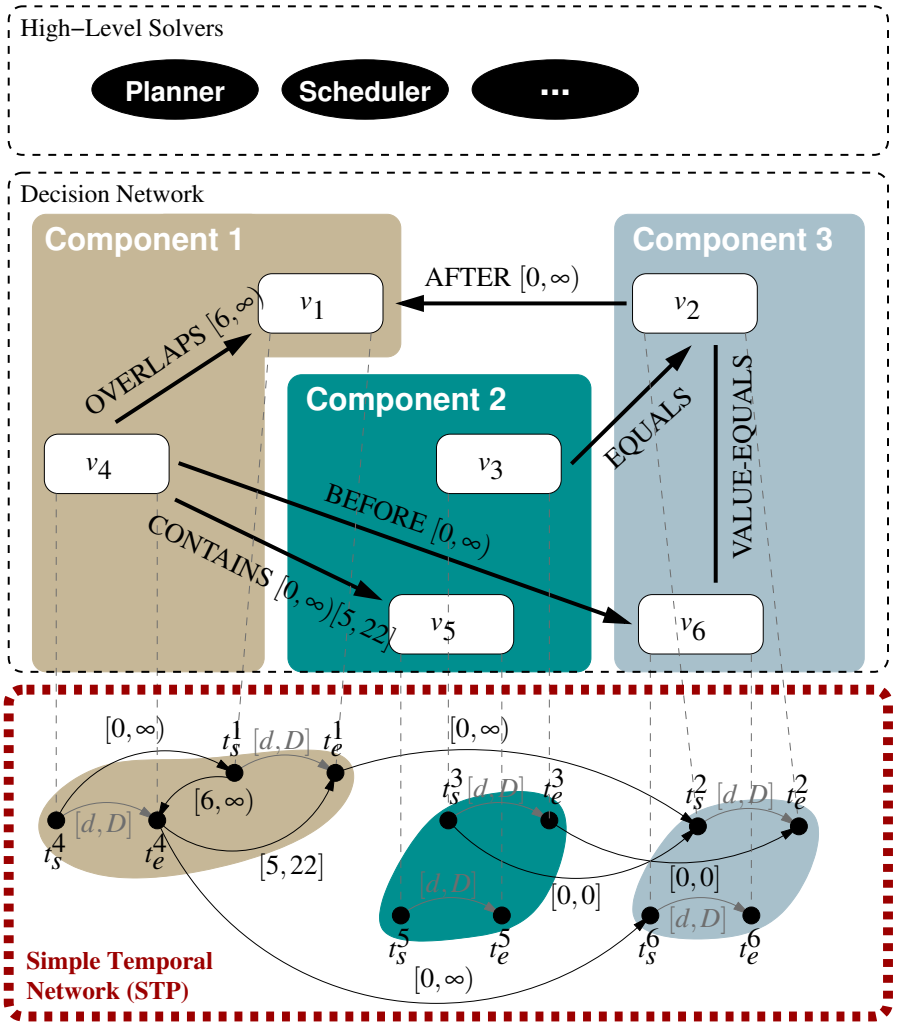
- OMPS is a three-tiered architecture in which the Decision Network allows to share information between tiers
- **DN represents the current state of computation**
- High-level solvers add/remove decisions and constraints to the DN
- OMPS provides a framework to implement new high-level reasoning modules

Constraint Reasoning in OMPS



- OMPS is a three-tiered architecture in which the Decision Network allows to share information between tiers
- DN represents the current state of computation
- **High-level solvers add/remove decisions and constraints to the DN**
- OMPS **provides a framework to implement new high-level reasoning modules**

Constraint Reasoning in OMPS

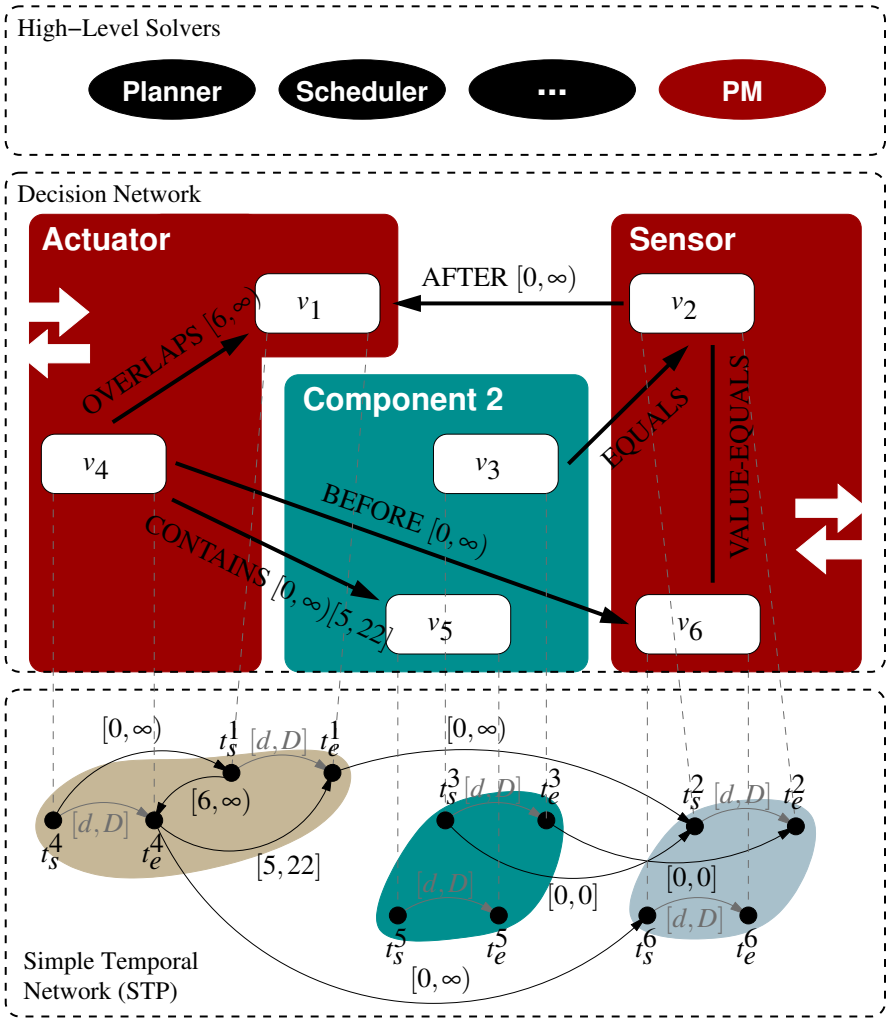


- Decisions' flexible time intervals represented as pairs of time points in a **Simple Temporal Problem (STP)** [Dechter et al., 1991]
- Each decision represented by **start** and **end time points in the STP**
- High-level temporal constraints translated to **simple distance constraints** between decisions' time points
  - propagation is $O(n^3)$, n = number of time points ($O(n^2)$ if incremental)

Constraint Reasoning in OMPS

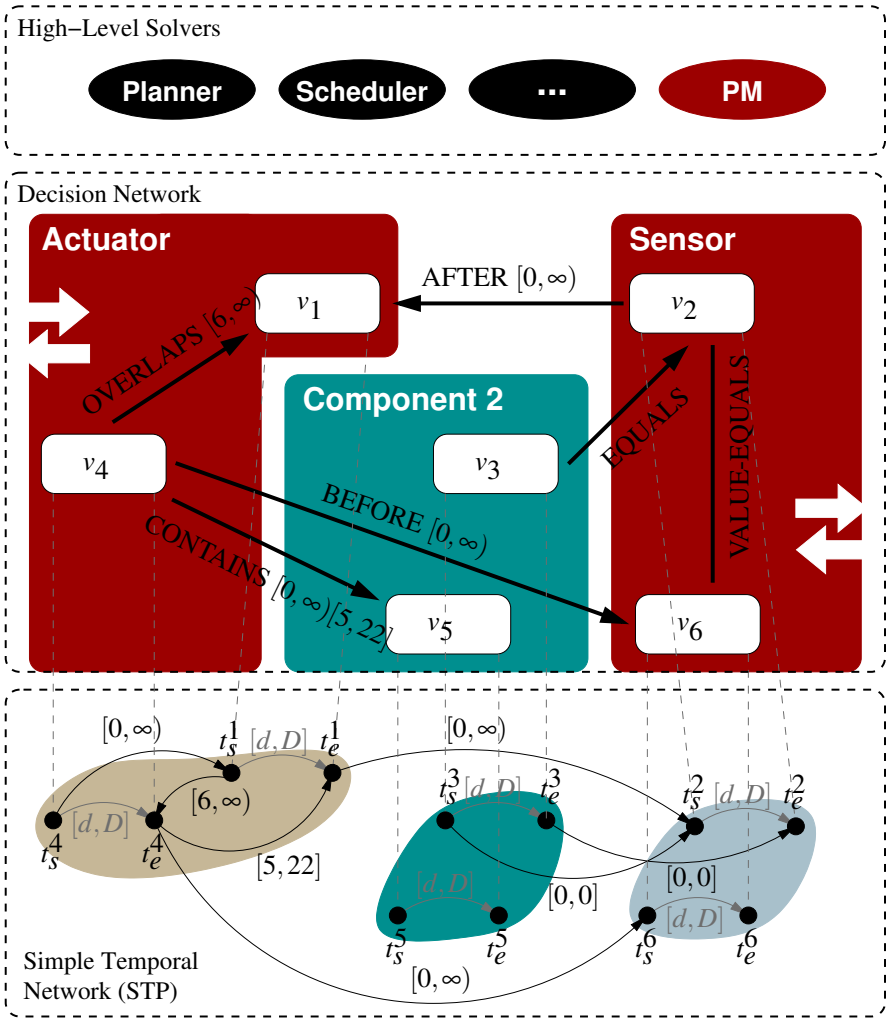
- OMPS provides a number of built-in functionalities
  1. propagation of temporal/value constraints
  2. suite of profile-based scheduling algorithms [Cesta et al., 2002, Policella et al., 2009]
  3. can deal with Disjunctive Temporal Problems at the lower layer [Oddi and Cesta, 2000]
  4. planning capabilities [Fratini et al., 2008]
  5. state variable and other component types
  6. ability to extend/create component types
- SAM leverages (4–6) to achieve an on-line activity monitoring system which is **fully integrated with the** PEIS**-Home**

SAM is an Activity Manager



- SAM is an Activity Manager = **Plan Monitor** + **Sensors** + **Actuators**
- The **Plan Monitor** is a **solver** that deduces possible activities in which the human is engaged in
- **Sensors** are **components** that represent the reality as it is observed in the real world in the DN
- **Actuators** are **components** that receive commands and maintain the execution status of real actuators

SAM is an Activity Manager



- SAM is an Activity Manager = **Plan Monitor** + **Sensors** + **Actuators**
- The **Plan Monitor** is a **solver** that deduces possible activities in which the human is engaged in
- **Sensors** are **components** that represent the reality as it is observed in the real world in the DN
- **Actuators** are **components** that receive commands and maintain the execution status of real actuators

The Plan Monitor

- The Plan Monitor **continuously** attempts to support all un-supported decisions in the DN
- This is done by a combination of **unification** and **domain theory expansion** steps [Fratini et al., 2008]

Unification

Assert that a decision occurs **simultaneously** with and has the **same value** as another decision

Domain Thoery Expansion

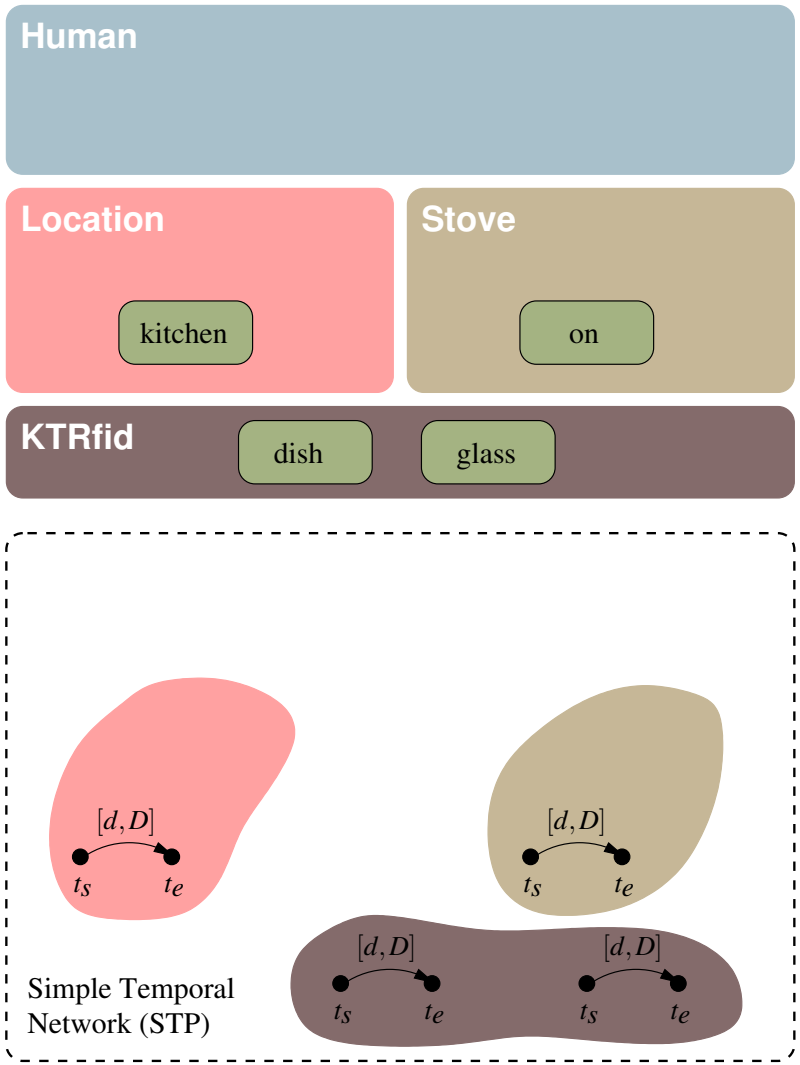
Assert the **consequences** of a decision on **other components**

The Plan Monitor – Unification

- Supporting a decision by unification ∼ "confirming" that a component is already due to take on that value in a given time frame
- Support by unification does not impose new decisions, **only constraints ensuring that support is present** in the DN

Querying sensors

If the DN represents real world sensor readings, then **unification is a way to "query" the sensors**

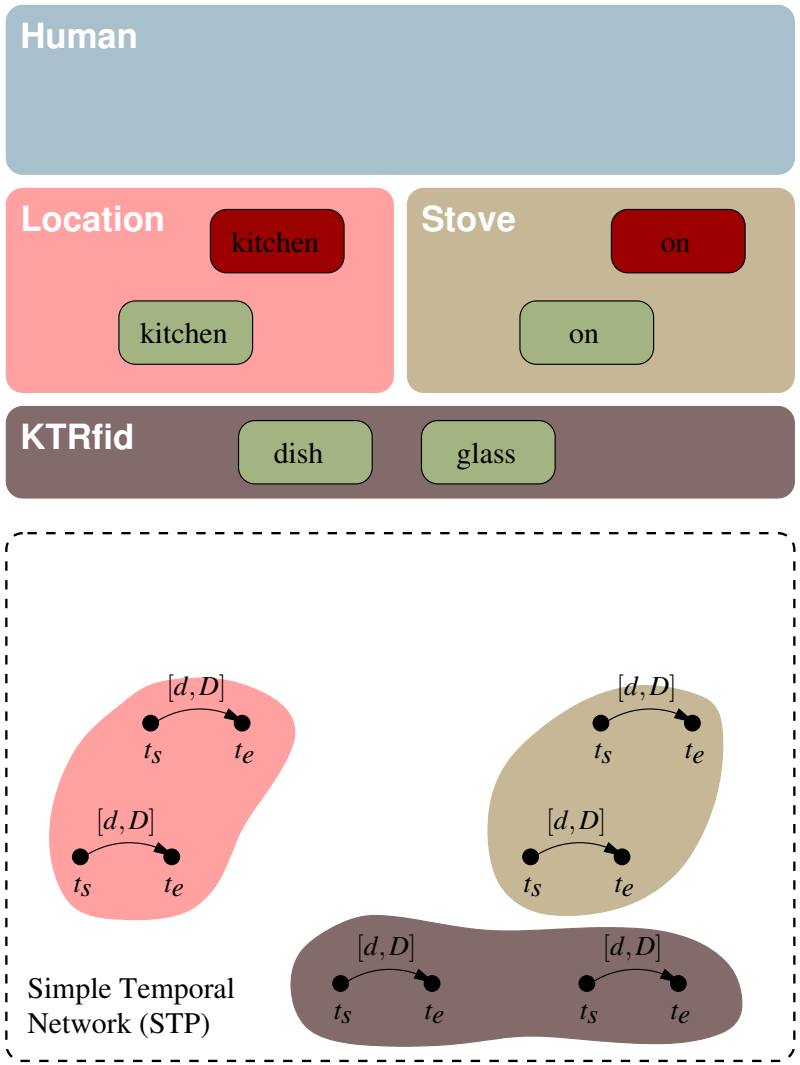


The Plan Monitor – Unification

- Supporting a decision by unification ∼ "confirming" that a component is already due to take on that value in a given time frame
- Support by unification does not impose new decisions, **only constraints ensuring that support is present** in the DN

Querying sensors

If the DN represents real world sensor readings, then **unification is a way to "query" the sensors**

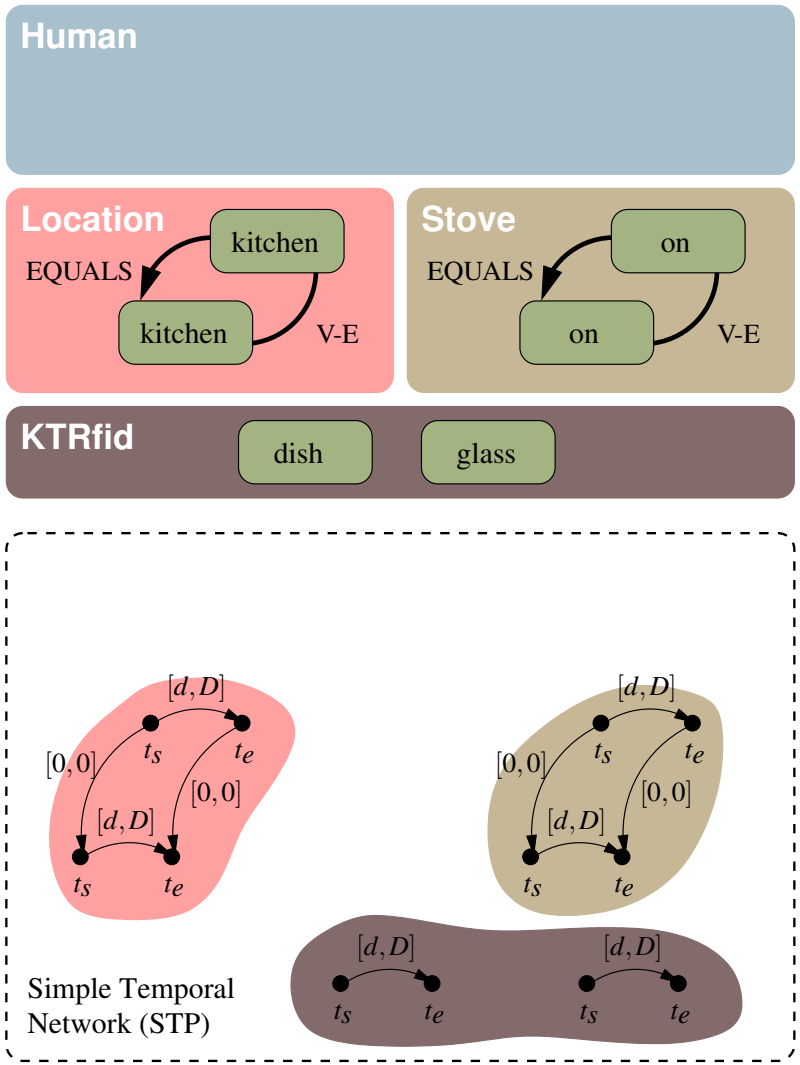


The Plan Monitor – Unification

- Supporting a decision by unification ∼ "confirming" that a component is already due to take on that value in a given time frame
- Support by unification does not impose new decisions, **only constraints ensuring that support is present** in the DN

Querying sensors

If the DN represents real world sensor readings, then **unification is a way to "query" the sensors**

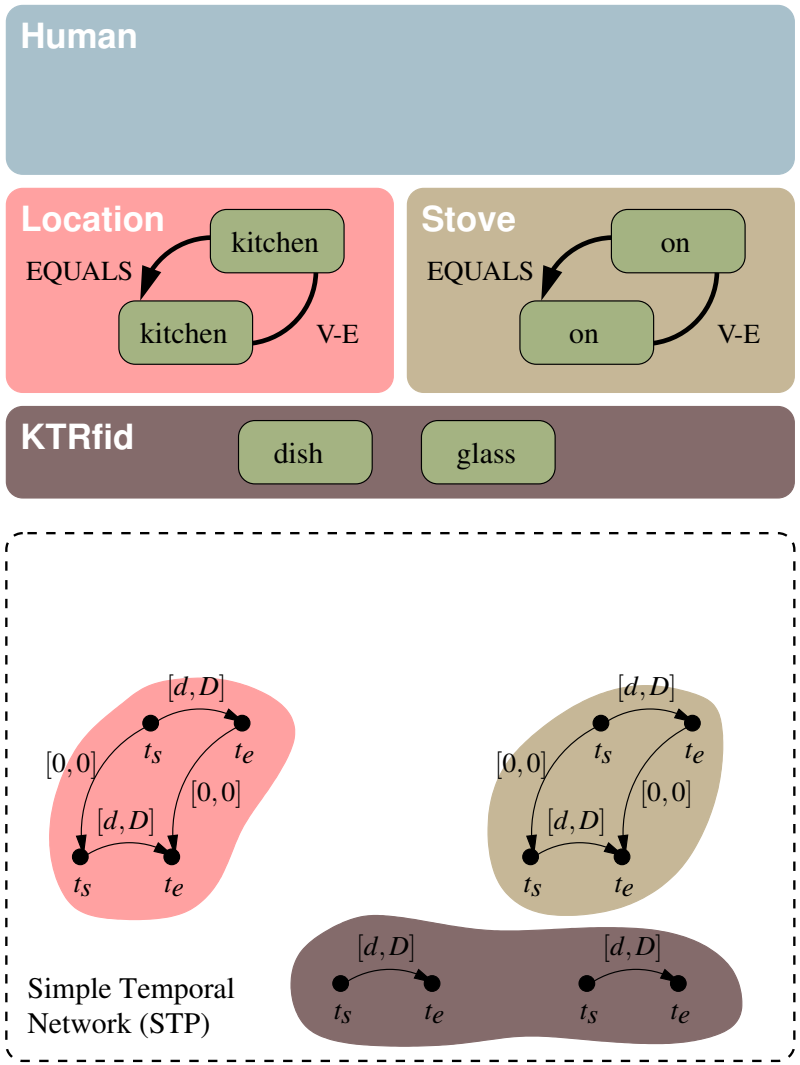


The Plan Monitor – Unification

- Supporting a decision by unification ∼ "confirming" that a component is already due to take on that value in a given time frame
- Support by unification does not impose new decisions, **only constraints ensuring that support is present** in the DN

Querying sensors

If the DN represents real world sensor readings, then **unification is a way to "query" the sensors**



The Plan Monitor – Domain Theory Expansion

- **Domain Theory:** a collection of synchronizations
- **Synchronization:** a collection of requirements on other components
- **Requirements:** sub-network of decisions to be added to other components in support of this decision

Human : **Cooking**
EQUALS Stove : **on**
DURING $[0,\infty)[0,\infty)$ Location : **kitchen**

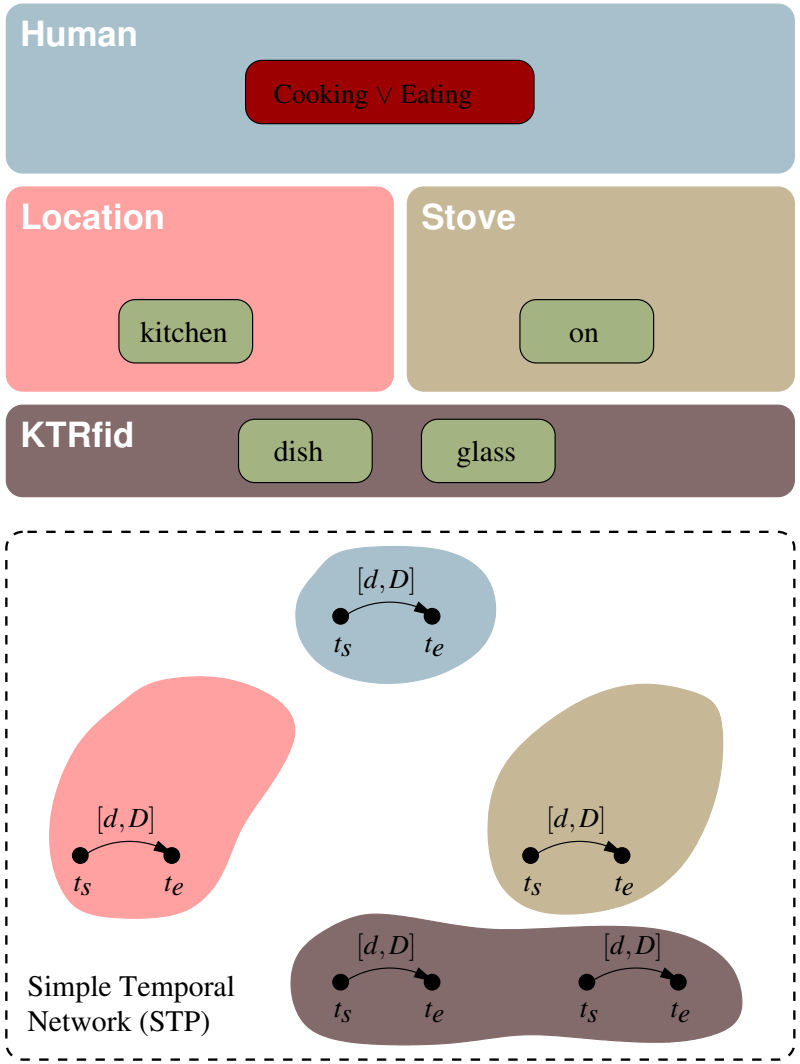
Human : **Eating**
EQUALS KTRfid : **dish**
DURING $[0,\infty)[0,\infty)$ Location : **kitchenTable**

The Plan Monitor – Domain Theory Expansion

- Supporting a decision by domain theory expansion ∼ finding "indirect" support for the decision through other components
- Support by expansion imposes **new decisions and constraints ensuring that support is present** in the DN

Activity recognition

If the DN represents real world sensor readings, then **expansion is a way to "hypothesize" a case for support for a given decision**

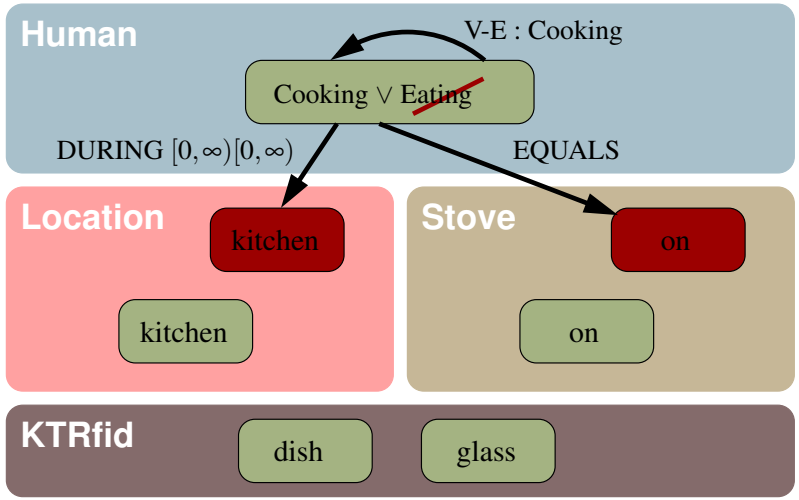


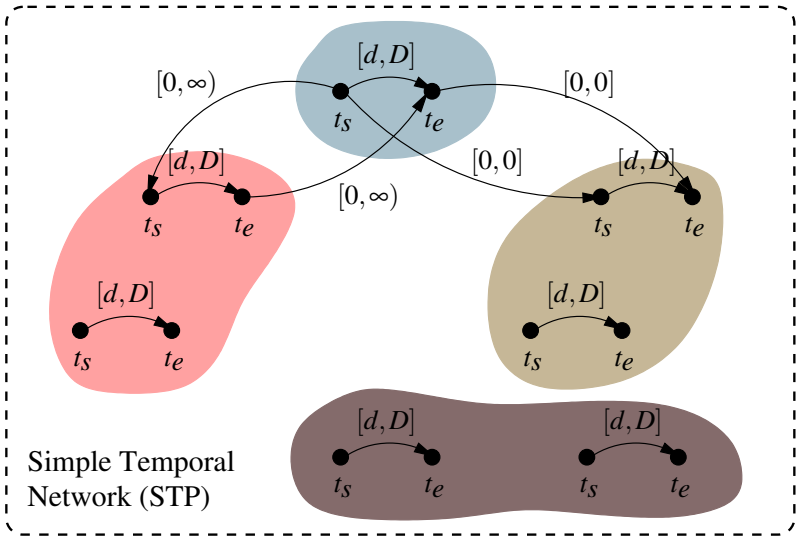
The Plan Monitor – Domain Theory Expansion

- Supporting a decision by domain theory expansion ∼ finding "indirect" support for the decision through other components
- Support by expansion imposes **new decisions and constraints ensuring that support is present** in the DN

Activity recognition

If the DN represents real world sensor readings, then **expansion is a way to "hypothesize" a case for support for a given decision**



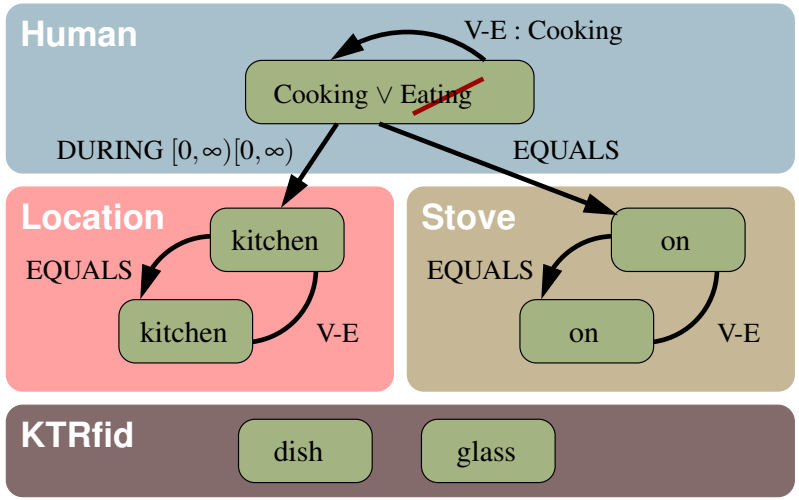


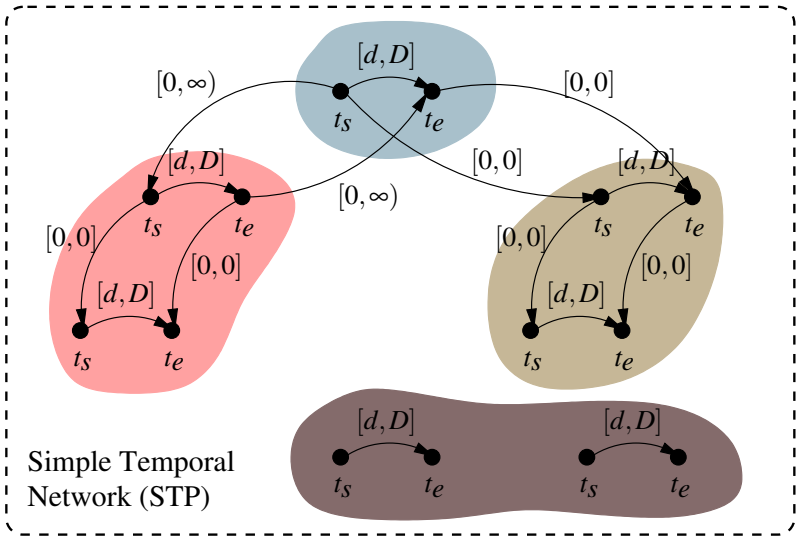
The Plan Monitor – Domain Theory Expansion

- Supporting a decision by domain theory expansion ∼ finding "indirect" support for the decision through other components
- Support by expansion imposes **new decisions and constraints ensuring that support is present** in the DN

Activity recognition

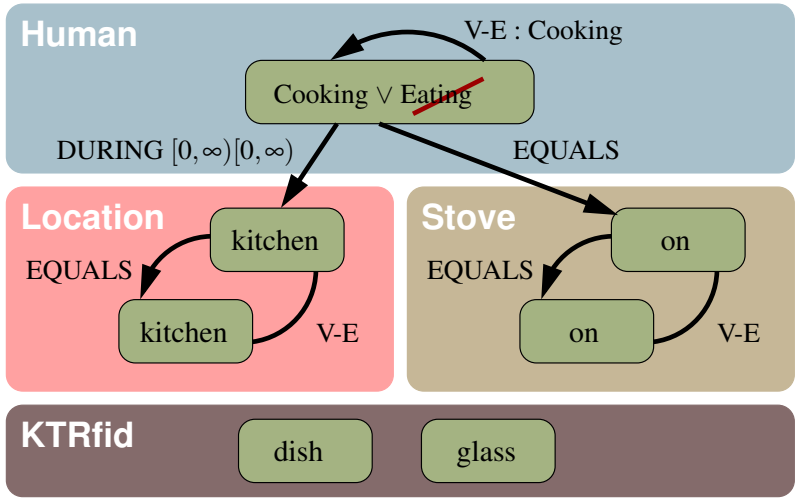
If the DN represents real world sensor readings, then **expansion is a way to "hypothesize" a case for support for a given decision**

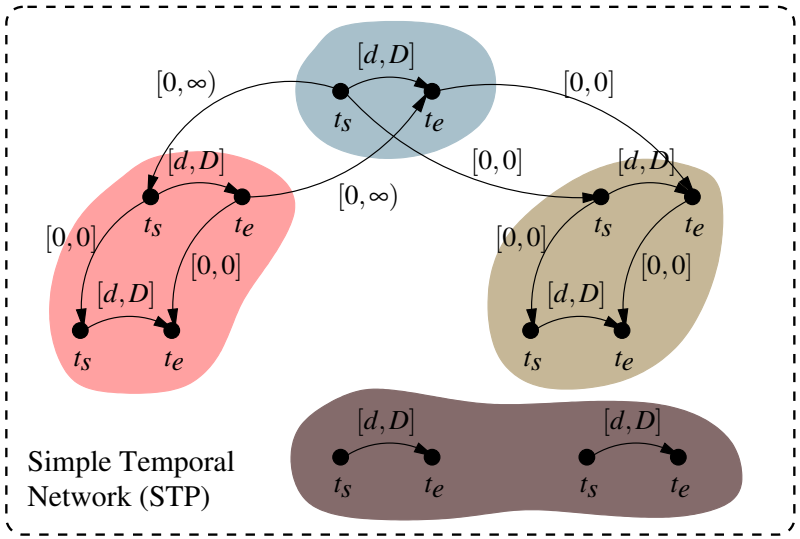




The Plan Monitor – Domain Theory Expansion

- Supporting a decision by domain theory expansion ∼ finding "indirect" support for the decision through other components
- Support by expansion imposes **new decisions and constraints ensuring that support is present** in the DN





Activity recognition

If the DN represents real world sensor readings, then **expansion is a way to "hypothesize" a case for support for a given decision**

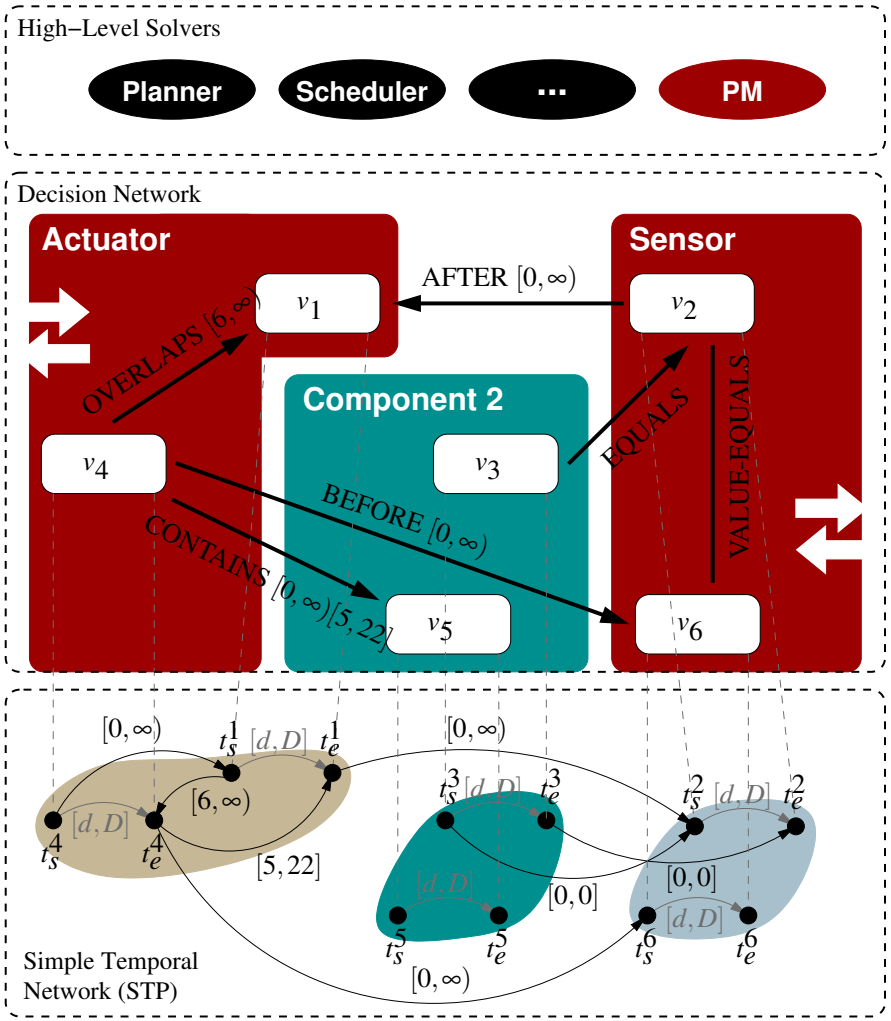
The Plan Monitor: Details

- The Plan Monitor continuously executes the following

Procedure `Replan(`*DN*`)`

foreach *component* c **do**
  if *c is controllable* **then**
    $\mathbf{v} = \bigvee_{v_i \in \text{possibleValues}(c)} v_i$
    DN $\leftarrow d = \langle \mathbf{v}, [I_s, I_e] \rangle$
    mark d as not supported
    foreach d' *on component* c **do** DN $\leftarrow d$ AFTER $[0, \infty)\, d'$
  `SupportDecisions(`*DN*`)`

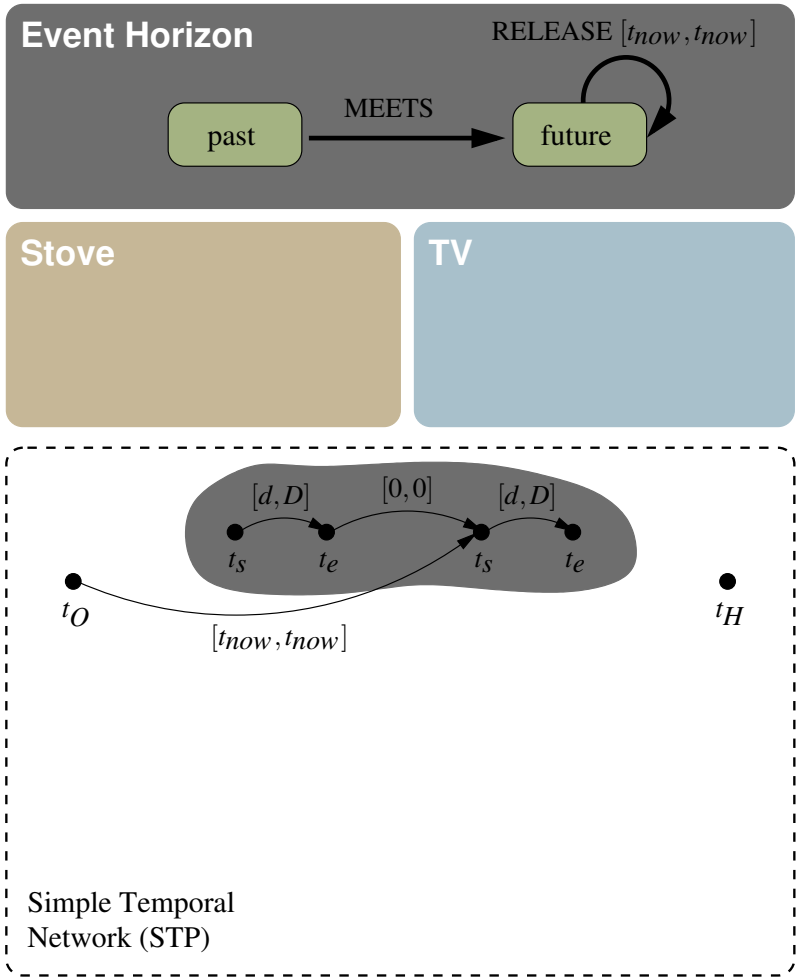
SAM is an Activity Manager



- SAM is an Activity Manager = **Plan Monitor** + **Sensors** + **Actuators**
- The **Plan Monitor** is a **solver** that deduces possible activities in which the human is engaged in
- **Sensors** are **components** that represent the reality as it is observed in the real world in the DN
- **Actuators** are **components** that receive commands and maintain the execution status of real actuators

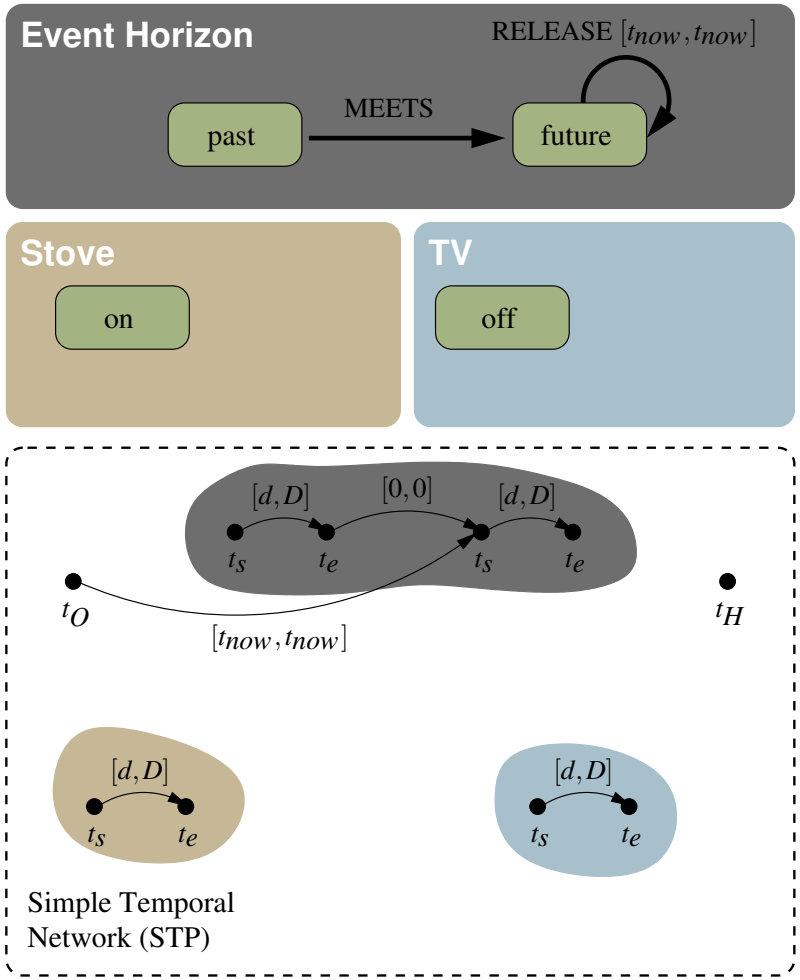
Sensor Components

- A special component represents **the current clock time** in the DN ("Event Horizon")
  - modifies the **release time constriant with current** t_{now}
- When a real-world sensor signals a new sensor reading, the SAM sensor
  - adds a **new decision**
  - constrains it to **start at** t_{now}
  - constrains it to **end in the future**



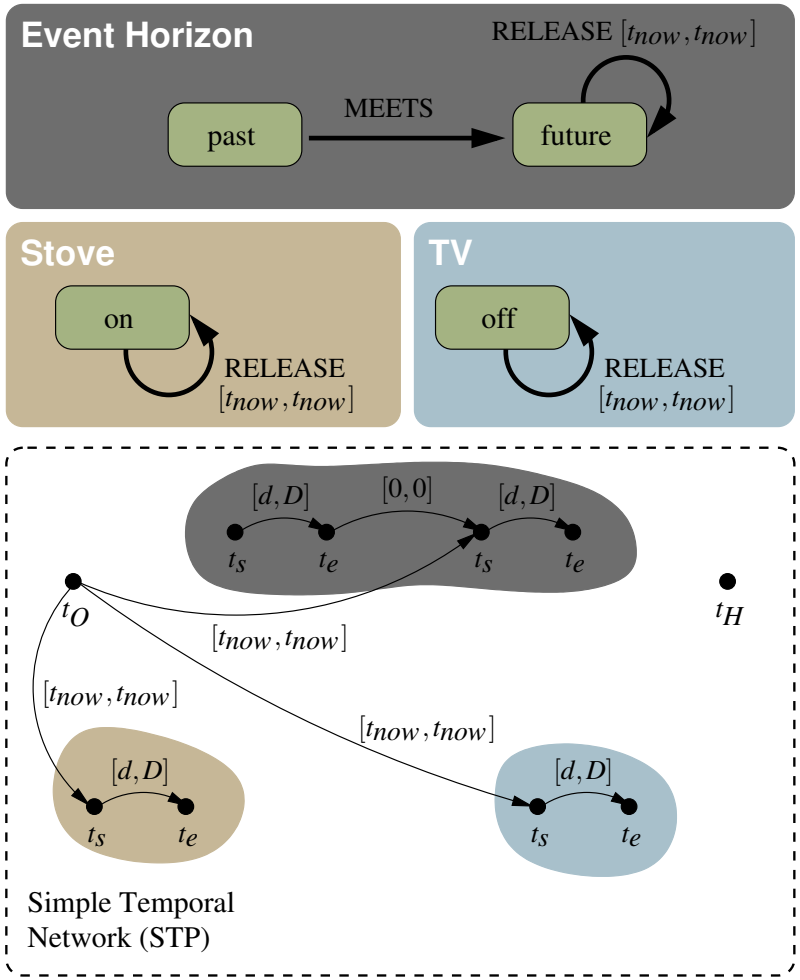
Sensor Components

- A special component represents **the current clock time** in the DN ("Event Horizon")
  - modifies the **release time constriant with current** t_{now}
- When a real-world sensor signals a new sensor reading, the SAM sensor
  - adds a **new decision**
  - constrains it to **start at** t_{now}
  - constrains it to **end in the future**



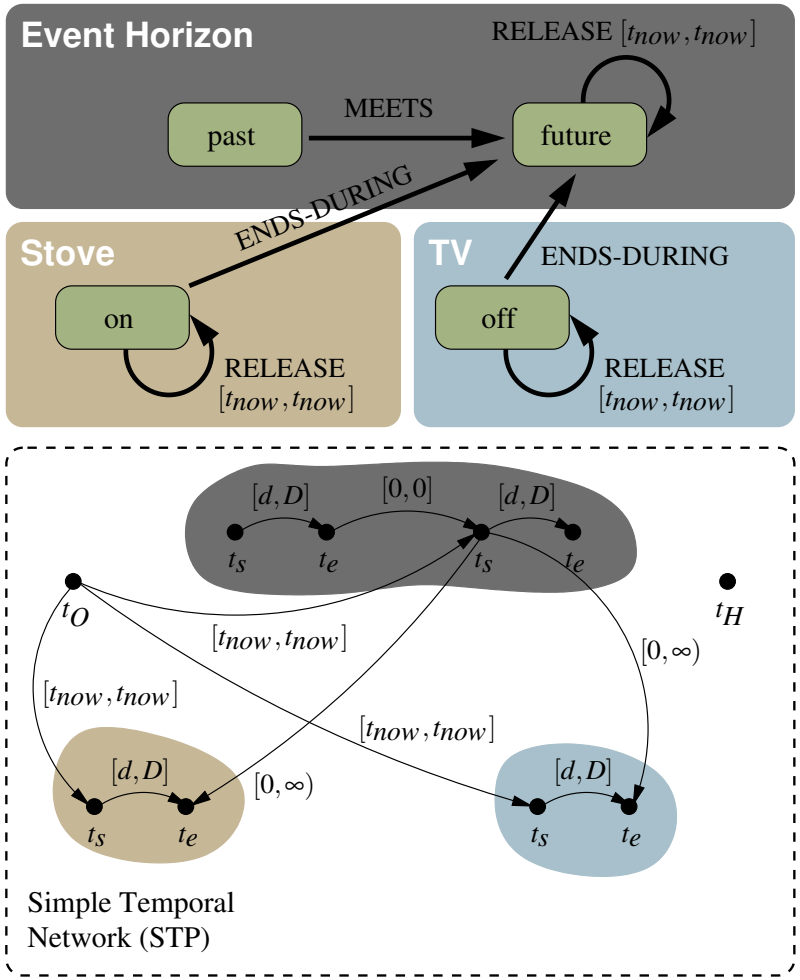
Sensor Components

- A special component represents **the current clock time** in the DN ("Event Horizon")
  - modifies the **release time constriant with current** t_{now}
- When a real-world sensor signals a new sensor reading, the SAM sensor
  - adds a **new decision**
  - constrains it to **start at** t_{now}
  - constrains it to **end in the future**



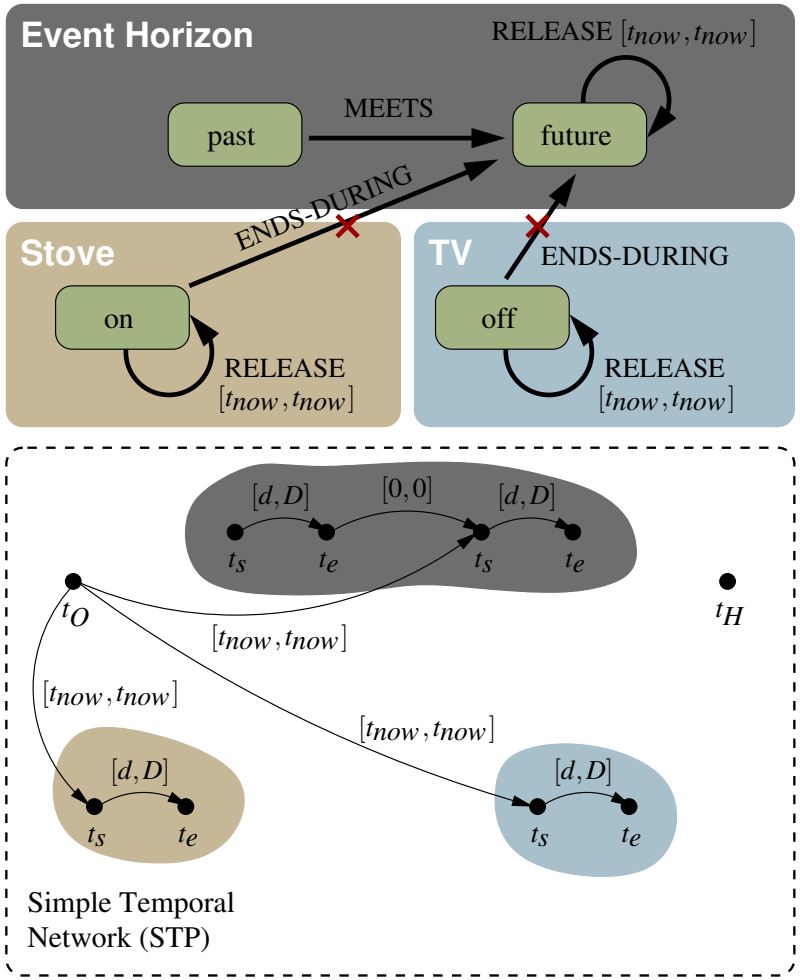
Sensor Components

- A special component represents **the current clock time** in the DN ("Event Horizon")
  - modifies the **release time constriant with current** t_{now}
- When a real-world sensor signals a new sensor reading, the SAM sensor
  - adds a **new decision**
  - constrains it to **start at** t_{now}
  - constrains it to **end in the future**



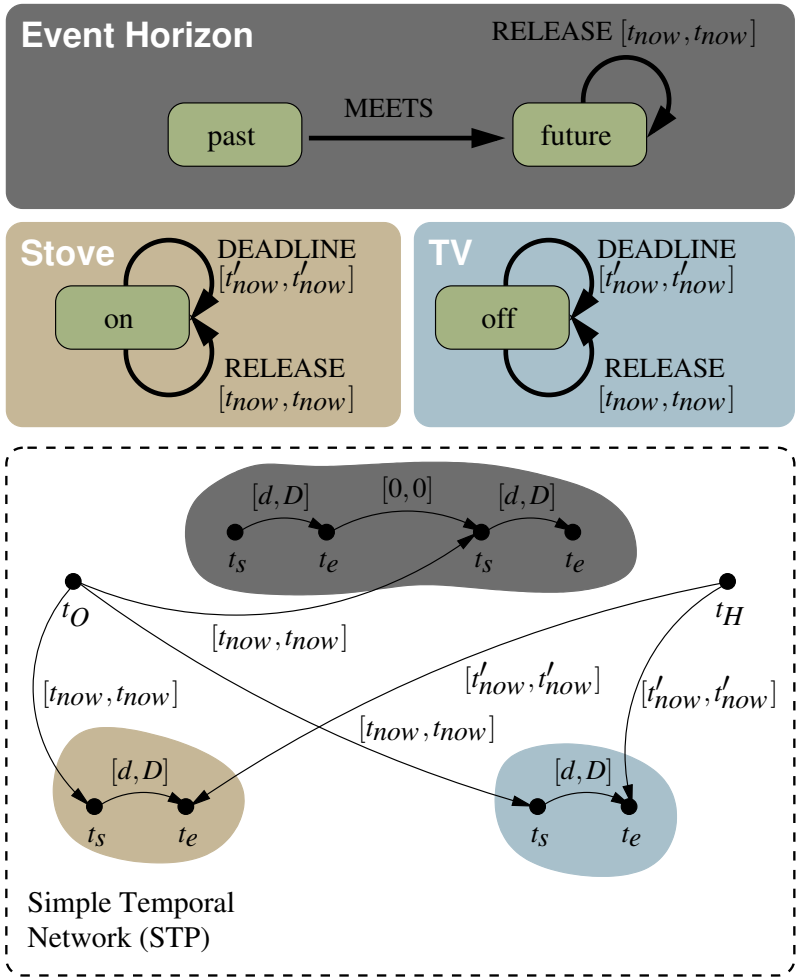
Sensor Components

- When a real-world sensor signals a that a sensor reading has changed, the SAM sensor
  - retracts **constraint with the "future" decision**
  - constrains it to **end at** t'_{now}
- Periodic updates to the release time of the "future" decision entails the **update of end times of current sensor readings**
- Propagation of sensor-related constraints entails the **update of the end times of recognized activities**



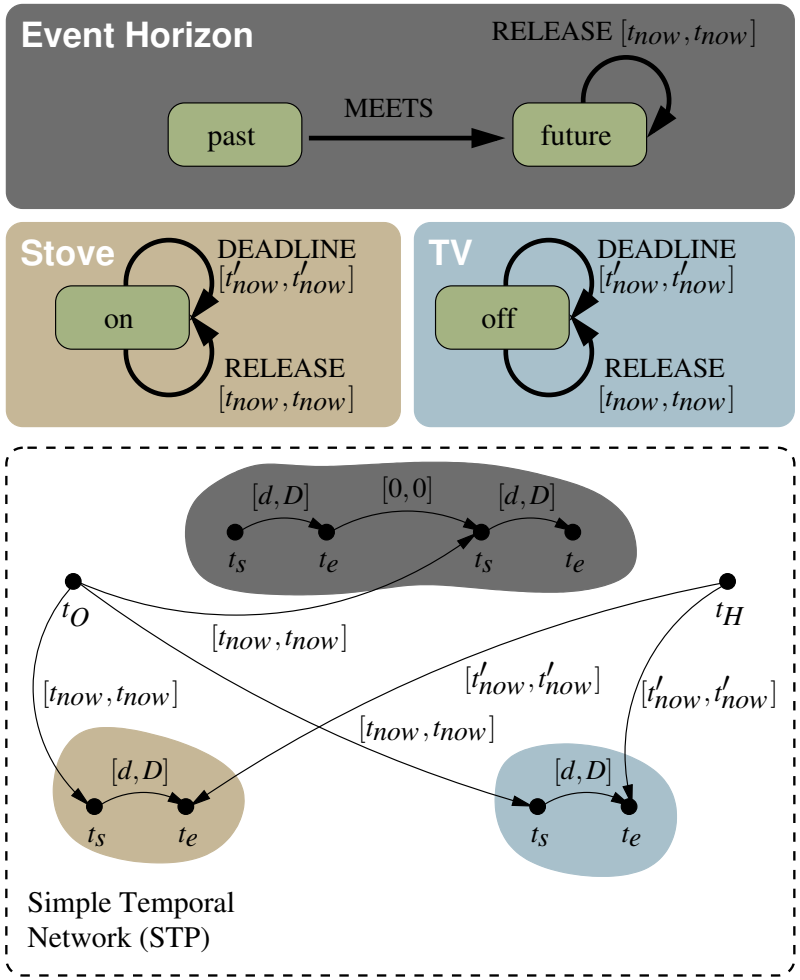
Sensor Components

- When a real-world sensor signals a that a sensor reading has changed, the SAM sensor
  - retracts **constraint with the "future" decision**
  - constrains it to **end at** t'_{now}
- Periodic updates to the release time of the "future" decision entails the **update of end times of current sensor readings**
- Propagation of sensor-related constraints entails the **update of the end times of recognized activities**



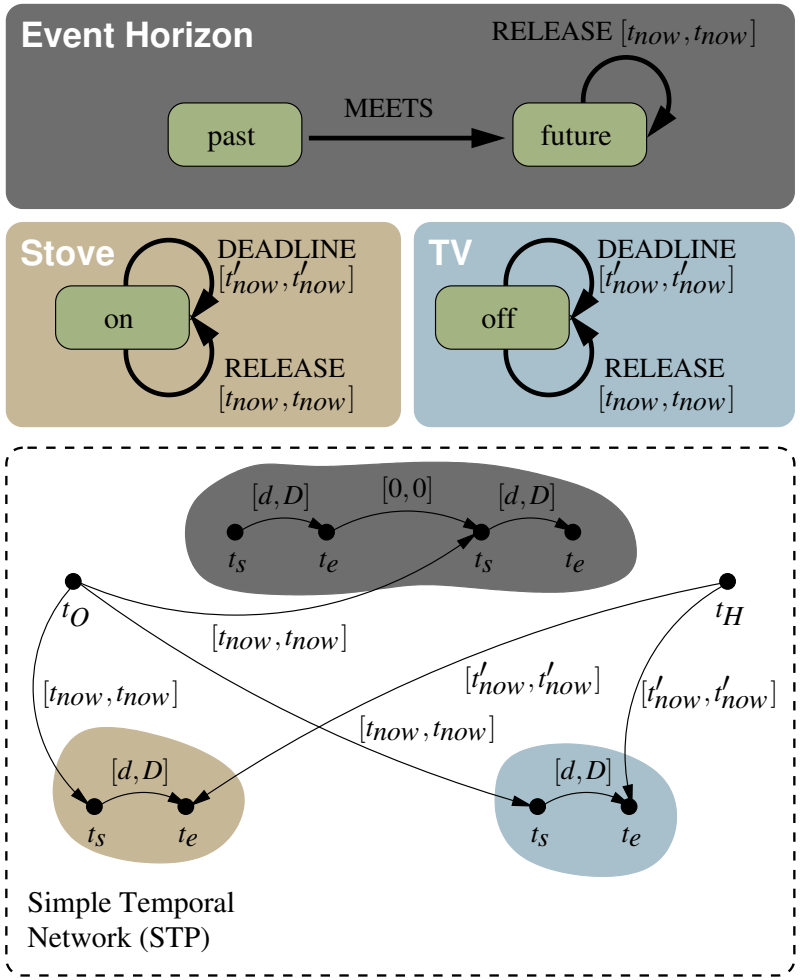
Sensor Components

- When a real-world sensor signals a that a sensor reading has changed, the SAM sensor
  - retracts **constraint with the "future" decision**
  - constrains it to **end at** t'_{now}
- Periodic updates to the release time of the "future" decision entails the **update of end times of current sensor readings**
- Propagation of sensor-related constraints entails the **update of the end times of recognized activities**



Sensor Components

- When a real-world sensor signals a that a sensor reading has changed, the SAM sensor
  - retracts **constraint with the "future" decision**
  - constrains it to **end at** t'_{now}
- Periodic updates to the release time of the "future" decision entails the **update of end times of current sensor readings**
- Propagation of sensor-related constraints entails the **update of the end times of recognized activities**



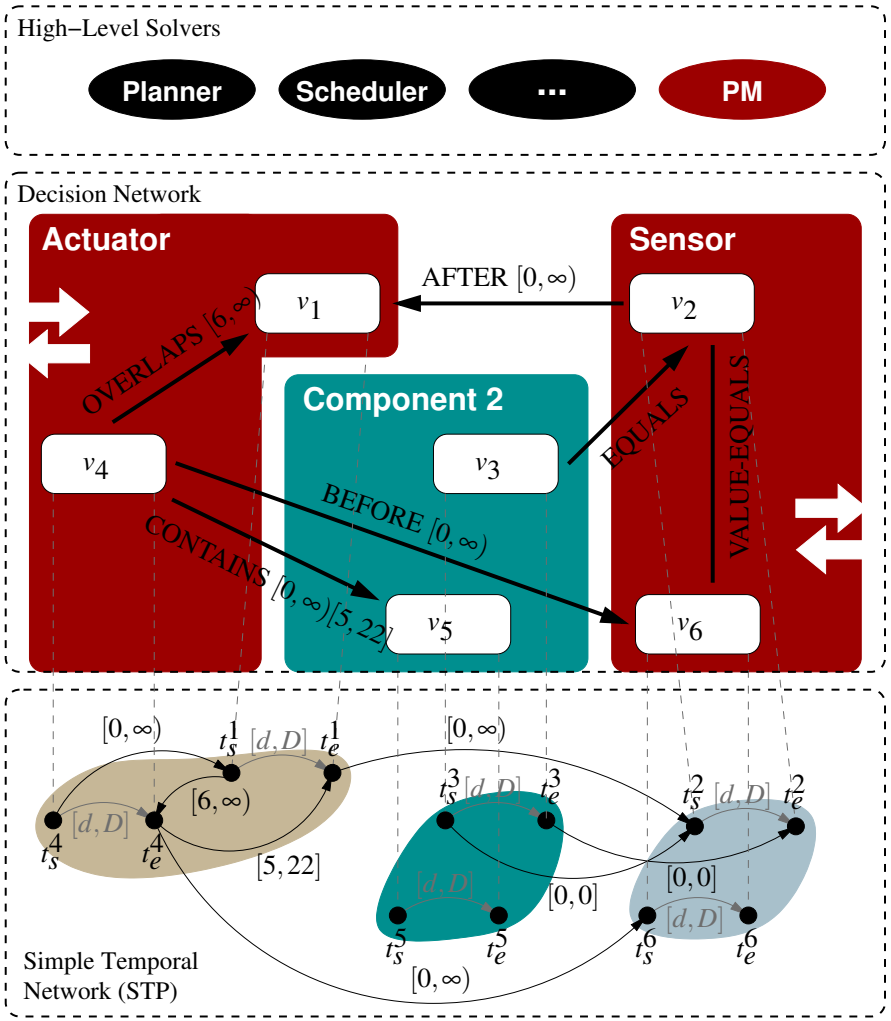
Sensor Components: Details

- Each sensor continuously executes the following

Procedure `UpdateSensorValues` (DN, t_{now})

$d = \langle \mathbf{v}, [[l_s, u_s], [l_e, u_e]] \rangle \in \text{DN} \;\; s.t. \;\; u_e = \infty$
$\mathbf{v_s} \leftarrow$ `ReadSensor()`
if $d = \text{null} \wedge \mathbf{v_s} \neq \text{null}$ **then**
    DN $\leftarrow$ $d' = \langle \mathbf{v_s}, [[0, \infty), [0, \infty)] \rangle$
    DN $\leftarrow$ d' RELEASE $[t_{\text{now}}, t_{\text{now}}]$
    DN $\leftarrow$ d' DEADLINE $[t_{\text{now}} + 1, \infty]$
else if $d \neq \text{null} \wedge \mathbf{v_s} = \text{null}$ **then** DN $\leftarrow$ d DEADLINE $[t_{\text{now}}, t_{\text{now}}]$
else if $d \neq \text{null} \wedge \mathbf{v_s} \neq \text{null}$ **then**
    if $\mathbf{v_s} = \mathbf{v}$ **then** DN $\leftarrow$ d DEADLINE $[t_{\text{now}}, \infty]$
    else
        DN $\leftarrow$ d DEADLINE $[t_{\text{now}}, t_{\text{now}}]$
        DN $\leftarrow$ $d' = \langle \mathbf{v_s}, [[0, \infty), [0, \infty)] \rangle$
        DN $\leftarrow$ d' RELEASE $[t_{\text{now}}, t_{\text{now}}]$
        DN $\leftarrow$ d' DEADLINE $[t_{\text{now}} + 1, \infty]$

SAM is an Activity Manager

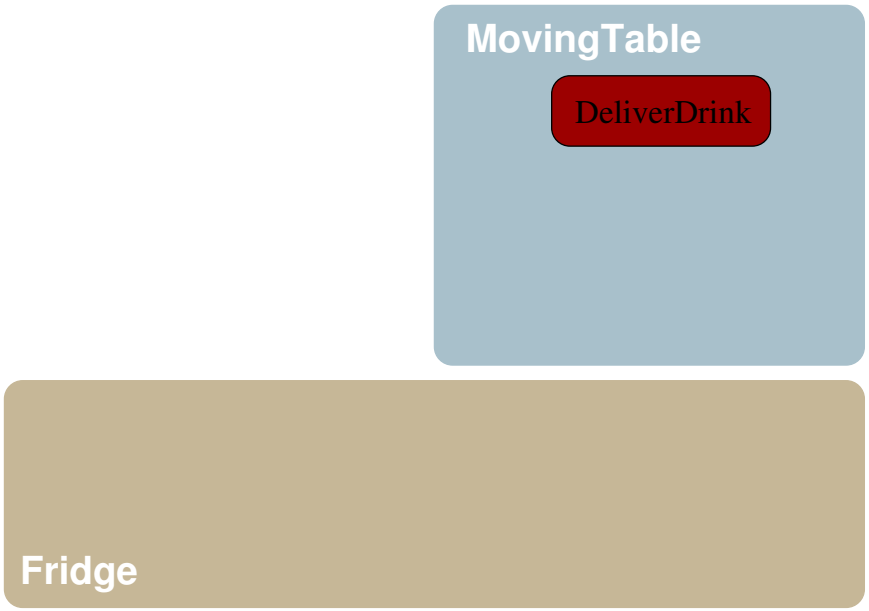


- SAM is an Activity Manager = **Plan Monitor** + **Sensors** + **Actuators**
- The **Plan Monitor** is a **solver** that deduces possible activities in which the human is engaged in
- **Sensors** are **components** that represent the reality as it is observed in the real world in the DN
- **Actuators** are **components** that receive commands and maintain the execution status of real actuators

Actuator Components

- **Actuators** trigger commands that are planned (i.e., that appear in the DN) on their physical counterparts
- Like sensors, actuators **maintain the status of execution** of planned decisions in the DN
  - "sense" whether a command has begun, is continuing, or has ended
  - add decisions, impose and retract constraints reflecting the perceived status
- An approach similar to [Jonsson et al., 2000]
- Details in forthcoming papers and [Cirillo et al., 2009]

Actuator Components: Example



MovingTable : **DockFridge**
MET-BY Fridge : **Open**

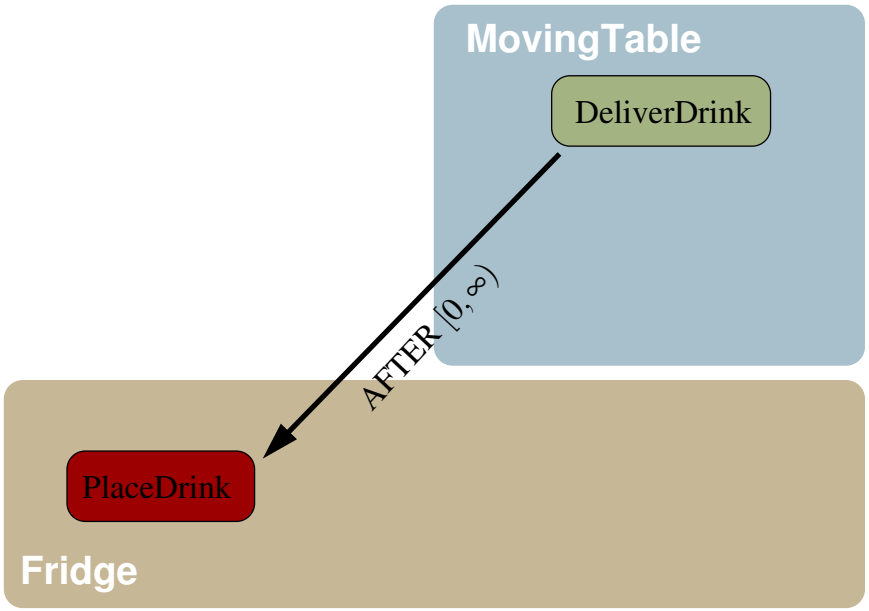
MovingTable : **UndockFridge**
BEFORE $[0, \infty)$ Fridge : **Close**

MovingTable : **DeliverDrink**
AFTER $[0, \infty)$ Fridge : **PlaceDrink**

Fridge : **PlaceDrink**
MET-BY MovingTable : **DockFridge**
MEETS MovingTable : **UnDockFridge**

Fridge : **Open**
MET-BY Fridge : **GraspDrink**

Actuator Components: Example



MovingTable : **DockFridge**
MET-BY Fridge : **Open**

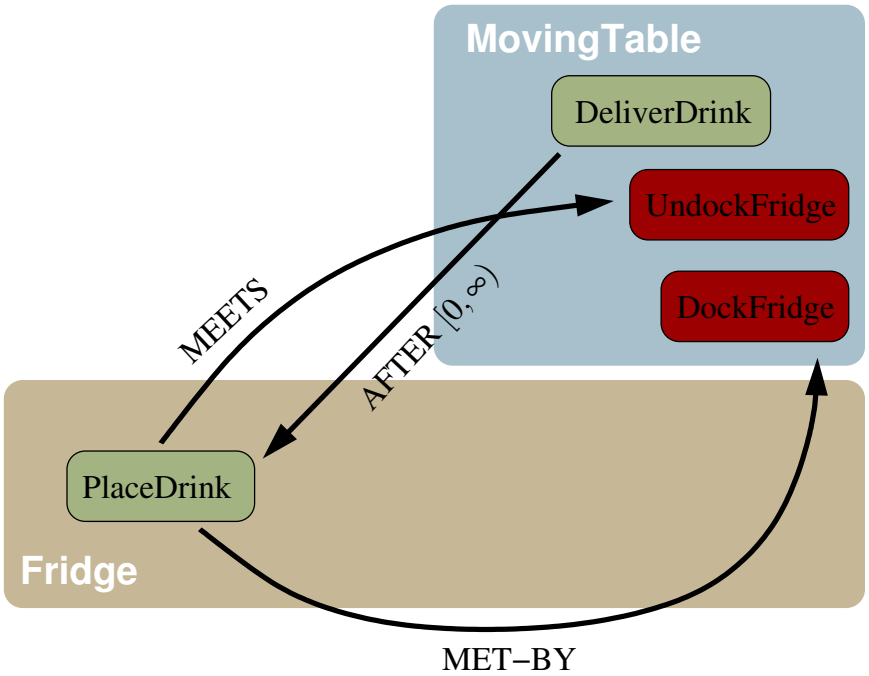
MovingTable : **UndockFridge**
BEFORE $[0, \infty)$ Fridge : **Close**

MovingTable : **DeliverDrink**
AFTER $[0, \infty)$ Fridge : **PlaceDrink**

Fridge : **PlaceDrink**
MET-BY MovingTable : **DockFridge**
MEETS MovingTable : **UnDockFridge**

Fridge : **Open**
MET-BY Fridge : **GraspDrink**

Actuator Components: Example



MovingTable : **DockFridge**
MET-BY Fridge : **Open**

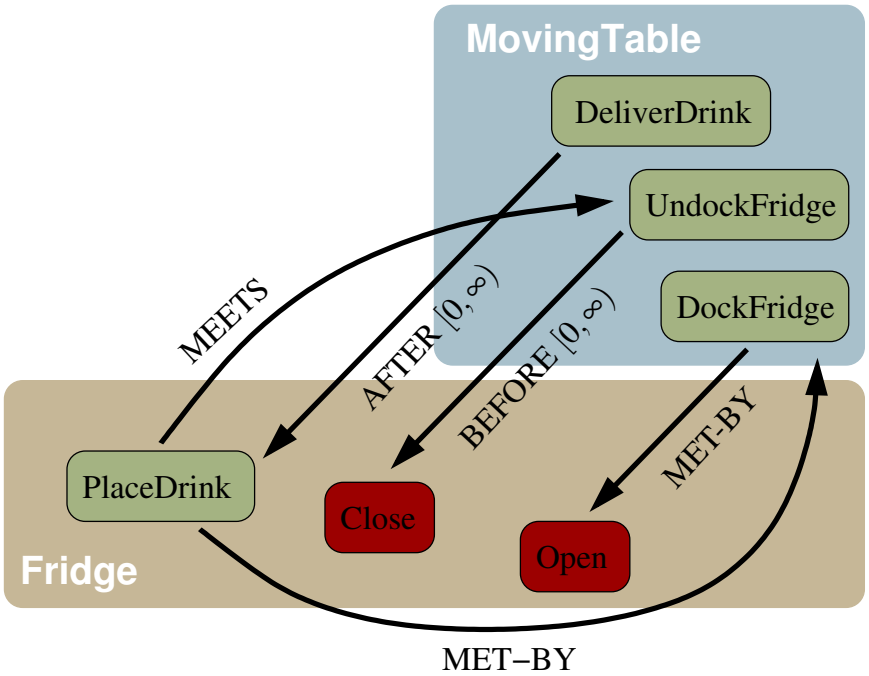
MovingTable : **UndockFridge**
BEFORE $[0,\infty)$ Fridge : **Close**

MovingTable : **DeliverDrink**
AFTER $[0,\infty)$ Fridge : **PlaceDrink**

Fridge : **PlaceDrink**
MET-BY MovingTable : **DockFridge**
MEETS MovingTable : **UnDockFridge**

Fridge : **Open**
MET-BY Fridge : **GraspDrink**

Actuator Components: Example



MovingTable : **DockFridge**
MET-BY Fridge : **Open**

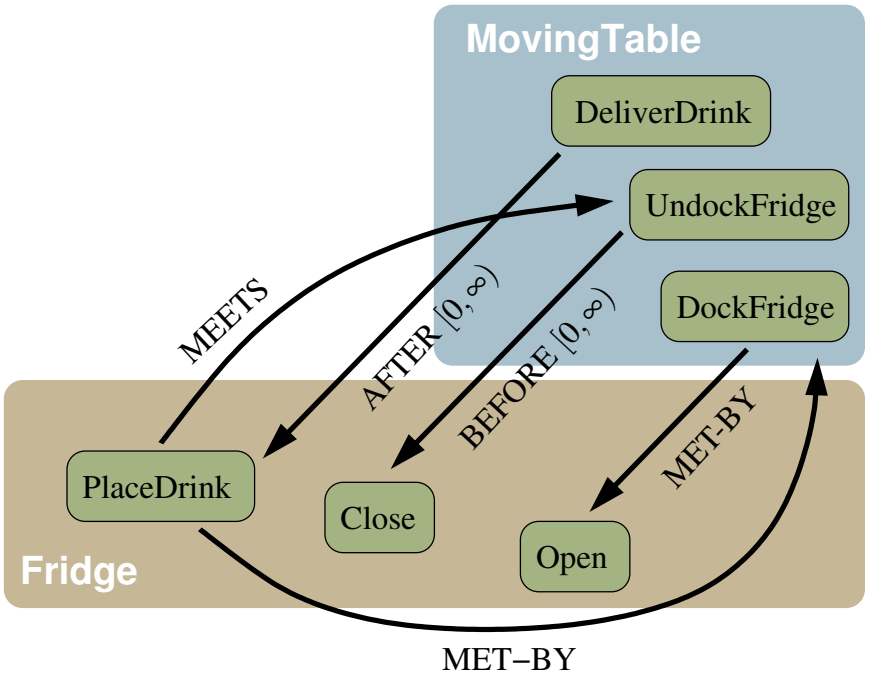
MovingTable : **UndockFridge**
BEFORE $[0,\infty)$ Fridge : **Close**

MovingTable : **DeliverDrink**
AFTER $[0,\infty)$ Fridge : **PlaceDrink**

Fridge : **PlaceDrink**
MET-BY MovingTable : **DockFridge**
MEETS MovingTable : **UnDockFridge**

Fridge : **Open**
MET-BY Fridge : **GraspDrink**

Actuator Components: Example



MovingTable : **DockFridge**
MET-BY Fridge : **Open**

MovingTable : **UndockFridge**
BEFORE $[0,\infty)$ Fridge : **Close**

MovingTable : **DeliverDrink**
AFTER $[0,\infty)$ Fridge : **PlaceDrink**

Fridge : **PlaceDrink**
MET-BY MovingTable : **DockFridge**
MEETS MovingTable : **UnDockFridge**

Fridge : **Open**
MET-BY Fridge : **GraspDrink**

Actuator Components: Details

- Each actuator continuously executes the following

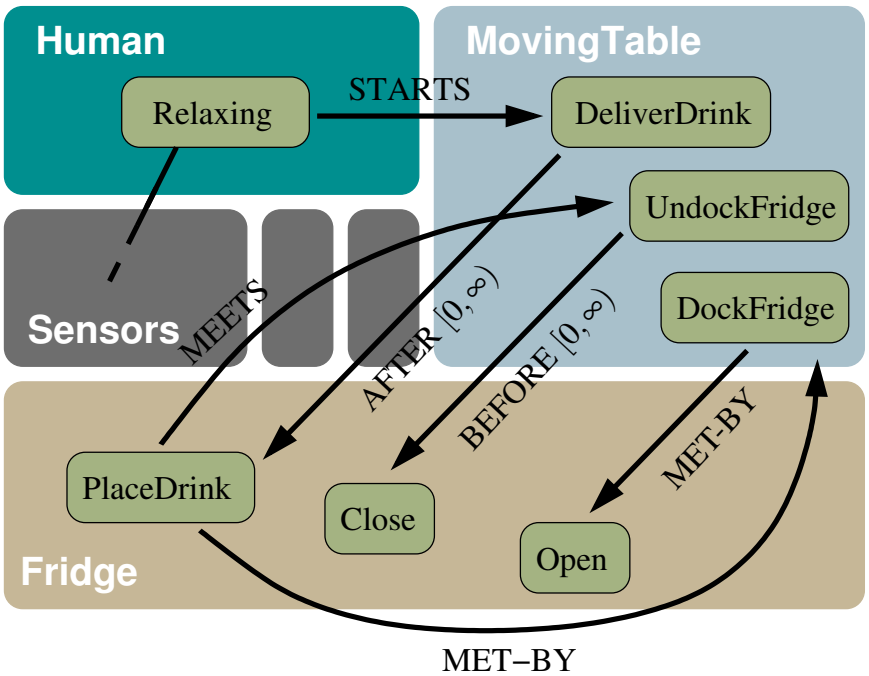
Procedure `UpdateExecutionState` (DN, t_{now})

$D = \{\langle \mathbf{v}, [[l_s, u_s], [l_e, u_e]] \rangle \in \text{DN} \ \ s.t. \ \ u_e = \infty\}$
foreach $d \in D$ **do**
    if `IsExecuting(v)` **then** DN $\leftarrow$ d DEADLINE $[t_{\text{now}} + 1, \infty]$
    else if $l_s = l_e$ **then** `StartExecuting(v)`
    DN $\leftarrow$ d RELEASE $[t_{\text{now}}, t_{\text{now}}]$
    else DN $\leftarrow$ d DEADLINE $[t_{\text{now}}, t_{\text{now}}]$

Recognition and Actuation in one Formalism

Both **activity recognition** and **actuation requirements** are modeled as **temporal relations among decisions**

Uniform Representation of Recognition and Actuation



Human : **Relaxing**
<requirements for recognition>
STARTS MovingTable : **DeliverDrink**

MovingTable : **DockFridge**
MET-BY Fridge : **Open**

MovingTable : **UndockFridge**
BEFORE $[0,\infty)$ Fridge : **Close**

MovingTable : **DeliverDrink**
AFTER $[0,\infty)$ Fridge : **PlaceDrink**

Fridge : **PlaceDrink**
MET-BY MovingTable : **DockFridge**
MEETS MovingTable : **UnDockFridge**

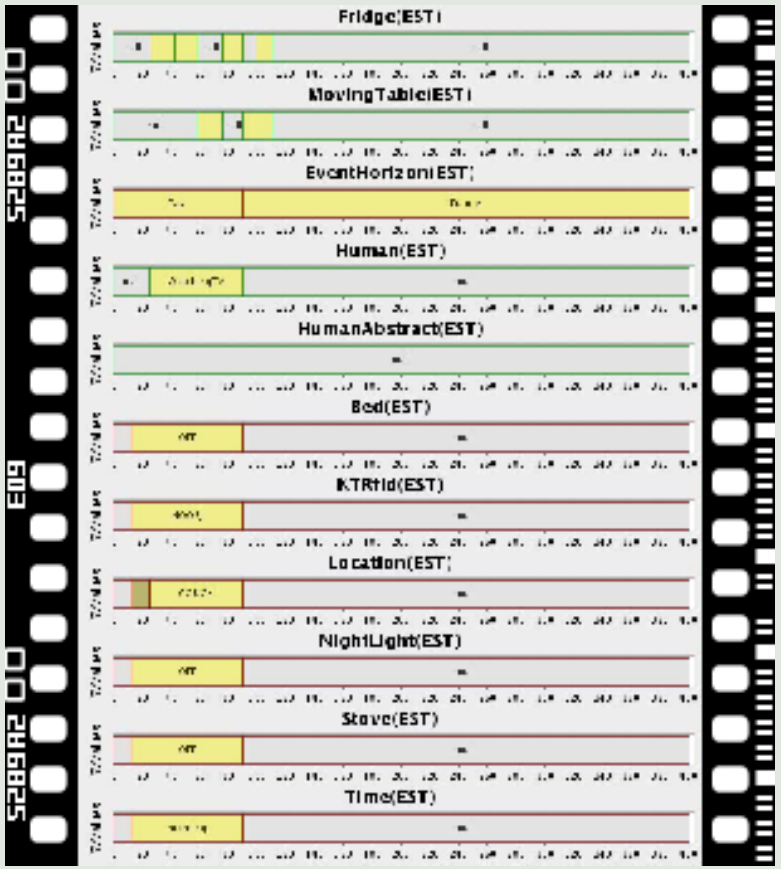
Fridge : **Open**
MET-BY Fridge : **GraspDrink**

Example Run

- SAM is interfaced with **five sensors** in the PEIS-Home
  - stereo camera for person localization
  - pressure sensor under the bed
  - RFID tag reader in the kitchen table and a number of tagged kitchen utensils
  - stove state sensor
  - luminosity sensor next to the bed
- **Two actuators** are also present
  - autonomous mobile table that can dock the fridge
  - actuated fridge that can grasp a drink and place it on the docked table
- A **human subject** carries out a number of actions in the PEIS-Home involving the use of the sensors

Example Run

The PEIS Home



SAM: Performance

- SAM heavily leverages temporal constraint propagation
  - complexity of adding/removing decisions and constraints to the DN is cubic in the number of decisions
- However, the bulk of computation is performed by the planner
  - still polynomial in the number of decisions (with a bounded horizon)
  - notice that number of decisions depends on how “active” the sensors are
- In a run like the previous, all processes can sample at a rate of about 1 Hz
  - we can recognize activities so long as their requirements persist within the sampling rate
- The performance of the plan monitor deteriorates for horizons of more than 20 minutes

Outline

Summary

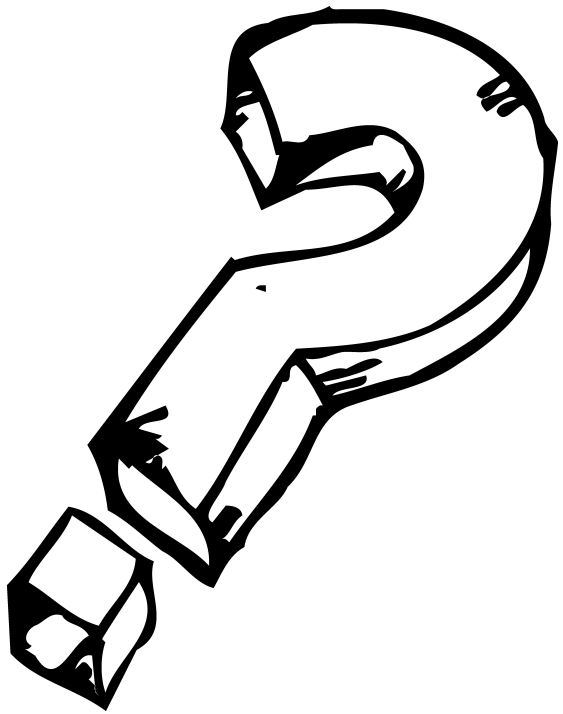
- SAM leverages temporal constraint reasoning to perform **concurrent activity reconition, planning and execution** in a sensor/actuator-rich environments
- Deductive, sensory and actuation processes **share information and reason on a dynamic constraint network**
  - sensors represent the **real world** as it is observed
  - actuators **trigger commands** and **monitor execution**
  - a plan monitor adds **deduced information on the context** of one or more monitored entities (e.g., a human user)
- Both context and actuation requirements are modeled as temporal relations among decisions
  - **single formalism** for recognition and actuation
- SAM is **fully integrated** into the PEIS-Home

SAM: Open Questions and On-Going Work

- How to increase **computational efficiency** of the system?
- How to ensure **robustness of recognition process**?
- How to include **plan forecasting** capability?
- How to model **preferences/soft requirements**?

See Appendix: "Open Questions and On-Going Work"

Questions



Outline

Computational Efficiency

- How to increase **computational efficiency** of the system? *(J. Ullberg)*
  - application domain is such that entire parts of the DN become useless with time
  - recognition of activities in the evening probably does not depend on sensor readings obtained in the morning or afternoon
  - restricting planning and temporal propagation to decisions within a sliding window
- Motivation: **sleep disorder diagnosis domain**, in which weeks of daily and nightly activities need to be monitored

Robustness in Activity Recognition

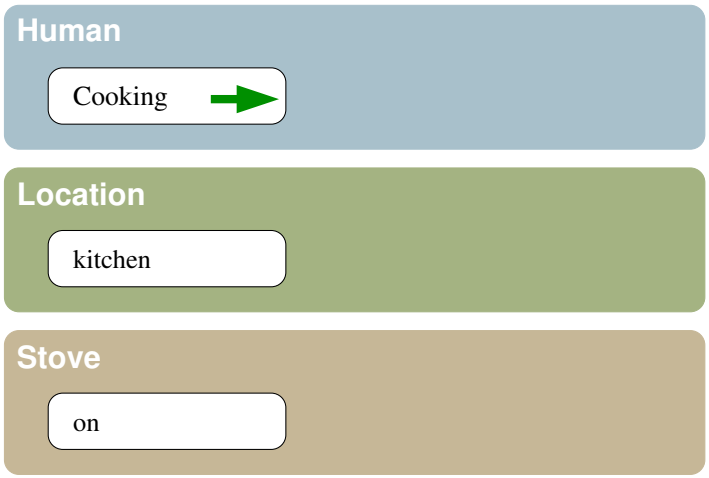
- How to ensure **robustness of recognition process**?
  (F. Dell'Osa, J. Ullberg)

Human : **Cooking**
EQUALS Stove : **on**
DURING $[0,\infty)[0,\infty)$ Location : **kitchen**

- Maintaining a different DN for each candidate outcome
- Late commitment based on analysis of constraint network

Robustness in Activity Recognition

- How to ensure **robustness of recognition process**?
  (F. Dell'Osa, J. Ullberg)

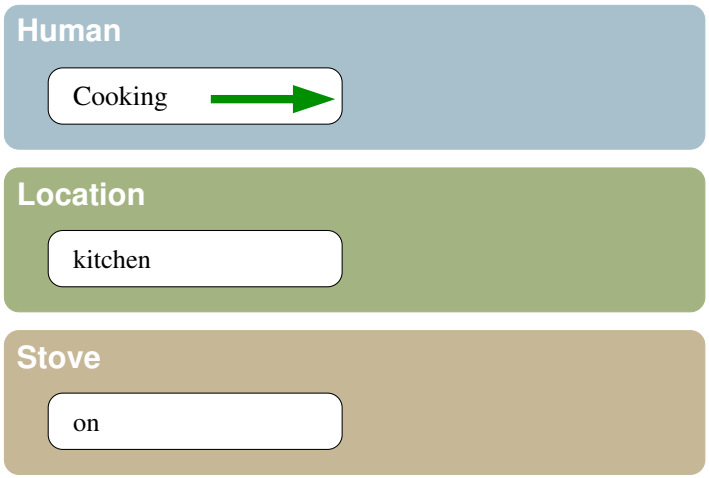


Human : **Cooking**
EQUALS Stove : **on**
DURING $[0,\infty)[0,\infty)$ Location : **kitchen**

- Maintaining a different DN for each candidate outcome
- Late commitment based on analysis of constraint network

Robustness in Activity Recognition

- How to ensure **robustness of recognition process**?
  (F. Dell'Osa, J. Ullberg)

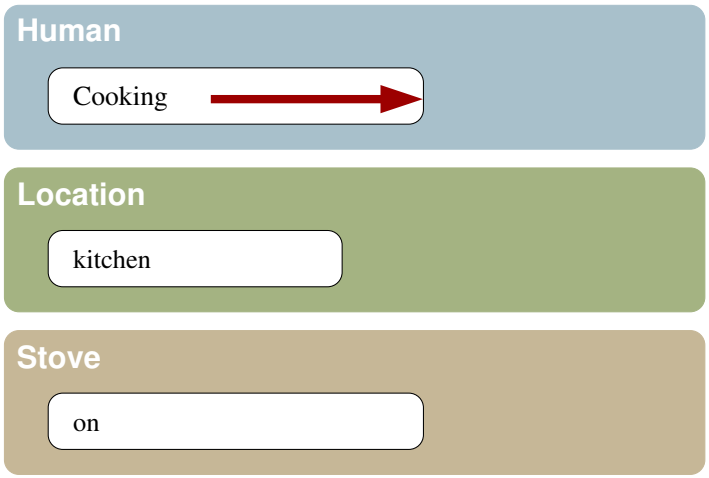


Human : **Cooking**
EQUALS Stove : **on**
DURING $[0,\infty)[0,\infty)$ Location : **kitchen**

- Maintaining a different DN for each candidate outcome
- Late commitment based on analysis of constraint network

Robustness in Activity Recognition

- How to ensure **robustness of recognition process**?
  (F. Dell'Osa, J. Ullberg)



Human : **Cooking**
EQUALS Stove : **on**
DURING $[0,\infty)[0,\infty)$ Location : **kitchen**

- Maintaining a different DN for each candidate outcome
- Late commitment based on analysis of constraint network

Plan Forecasting and Soft Requirements

- How to include **plan forecasting** capability?
  (M. Cirillo)
  - ability to forecast human plans can increase proactiveness of the system
  - instrumental for human-aware robot planning [Cirillo et al., 2008]
- How to model **preferences/soft requirements**?
  (F. Venturini)
  - adding the capability to reason about fuzzy temporal constraints
  - fuzzy extension of Allen's interval algebra [Badaloni and Giacomin, 2000], fuzzy extension of the Simple Temporal Problem [Marín et al., 1997]

Outline

Synergy with Other Approahces to Recognition

- **Data-driven approaches**: e.g., Hidden Markov Models (HMMs) for learning sequences of sensor observations with given transition probabilities (e.g., [Wu et al., 2007])
- **Knowledge-driven approaches**: patterns of observations are modeled from first principles rather than learned
- The two strategies seem **complementary in scope**
  - data-driven approaches provide an effective way to recognize elementary activities from large amounts of continuous data
  - knowledge-driven approaches are useful when the criteria for recognizing human activities are given by complex but general rules that are clearly identifiable
- Data-driven approaches have been **more extensively validated** within real scenarios
  - important to assess our approach in a real application domain

Related Work in Constraint Reasoning for Recognition and Execution

- Schedule execution monitoring techniques for domestic activity monitoring [Cesta et al., 2007, Pollack et al., 2003]
  - pre-compiled (albeit highly flexible) schedules as models for human behavior
  - SAM employs a planning process to actually instantiate such candidate schedules on-line
- Other timeline-based approaches to planning, e.g., [Jonsson et al., 2000]
  - address concurrent planning and execution, not specifically geared towards activity recognition
  - provides only state variable component type, lack of extensible component types (e.g., “active” components like sensors and actuators)

References


Allen, J. (1984).
Towards a general theory of action and time.
Artificial Intelligence, 23(2):123–154.


Badaloni, S. and Giacomin, M. (2000).
A fuzzy extension of Allen's interval algebra.
In Verlag, S., editor, *LNAI*, volume 1792, pages 1555–165.


Cesta, A., Cortellessa, G., Giuliani, M., Pecora, F., Scopelliti, M., and Tiberio, L. (2007).
Caring About the User's View: The Joys and Sorrows of Experiments with People.
In *ICAPS07 Workshop on Moving Planning and Scheduling Systems into the Real World*.


Cesta, A. and Fratini, S. (2008).
The Timeline Representation Framework as a Planning and Scheduling Software Development Environment.
In *Proc. of 27th Workshop of the UK Planning and Scheduling SIG*.


Cesta, A., Fratini, S., Oddi, A., and Pecora, F. (2008).
APSI Case#1: Pre-planning Science Operations in Mars Express.
In *Proc. of iSAIRAS-08*.


Cesta, A., Oddi, A., and Smith, S. F. (2002).
A constraint-based method for project scheduling with time windows.
Journal of Heuristics, 8(1):109–136.


Cirillo, M., Karlsson, L., and Saffiotti, A. (2008).
A framework for human-aware robot planning.
In *Proc. of the Scandinavian Conf. on Artificial Intelligence (SCAI)*, Stockholm, SE.

Online at http://www.aass.oru.se/~asaffio/.

Cirillo, M., Lanzellotto, F., Pecora, F., and Saffiotti, A. (2009).
Monitoring domestic activities with temporal constraints and components.
In *5th International Conference on Intelligent Environments (IE'09)*.
(to appear).

Dechter, R., Meiri, I., and Pearl, J. (1991).
Temporal constraint networks.
Artif. Intell., 49(1-3):61–95.

Fratini, S., Pecora, F., and Cesta, A. (2008).
Unifying Planning and Scheduling as Timelines in a Component-Based Perspective.
Archives of Control Sciences, 18(2):231–271.

Jonsson, A., Morris, P., Muscettola, N., Rajan, K., and Smith, B. (2000).
Planning in Interplanetary Space: Theory and Practice.
In *Proc. Int. Conf. on AI Planning and Scheduling (AIPS-00)*.

Marín, R., Cárdenas, M., Balsa, M., and Sánchez, J. L. (1997).
Obtaining solutions in fuzzy constraint networks.
International Journal of Approximate Reasoning, 16:261–288.

Oddi, A. and Cesta, A. (2000).
Incremental forward checking for the disjunctive temporal problem.
In *Proceedings of ECAI*.

Policella, N., Cesta, A., Oddi, A., and Smith, S. (2009).
Solve-and-Robustify. Synthesizing Partial Order Schedules by Chaining.
Journal of Scheduling, 12(3):299–314.

Pollack, M., Brown, L., Colbry, D., McCarthy, C., Orosz, C., Peintner, B., Ramakrishnan, S., and Tsamardinos, I. (2003).

Autominder: an intelligent cognitive orthotic system for people with memory impairment.
Robotics and Autonomous Systems, 44(3-4):273–282.

Saffiotti, A., Broxvall, M., Gritti, M., LeBlanc, K., Lundh, R., J., R., Seo, B., and Cho, Y. (2008).
The PEIS-ecology project: vision and results.
In *Proc of the IEEE/RSJ Int Conf on Intelligent Robots and Systems (IROS)*, Nice, France.

Wu, J., Osuntogun, A., Choudhury, T., Philipose, M., and Rehg, J. (2007).
A Scalable Approach to Activity Recognition Based on Object Use.
In *Proceedings of ICCV 2007. Rio de Janeiro, Brazil*.