Programming Autonomous Behaviour Abstractions and Techniques

Felipe Rech Meneguzzi
`felipe.meneguzzi@pucrs.br`
Pontifícia Universidade Católica do Rio Grande do Sul

Overview

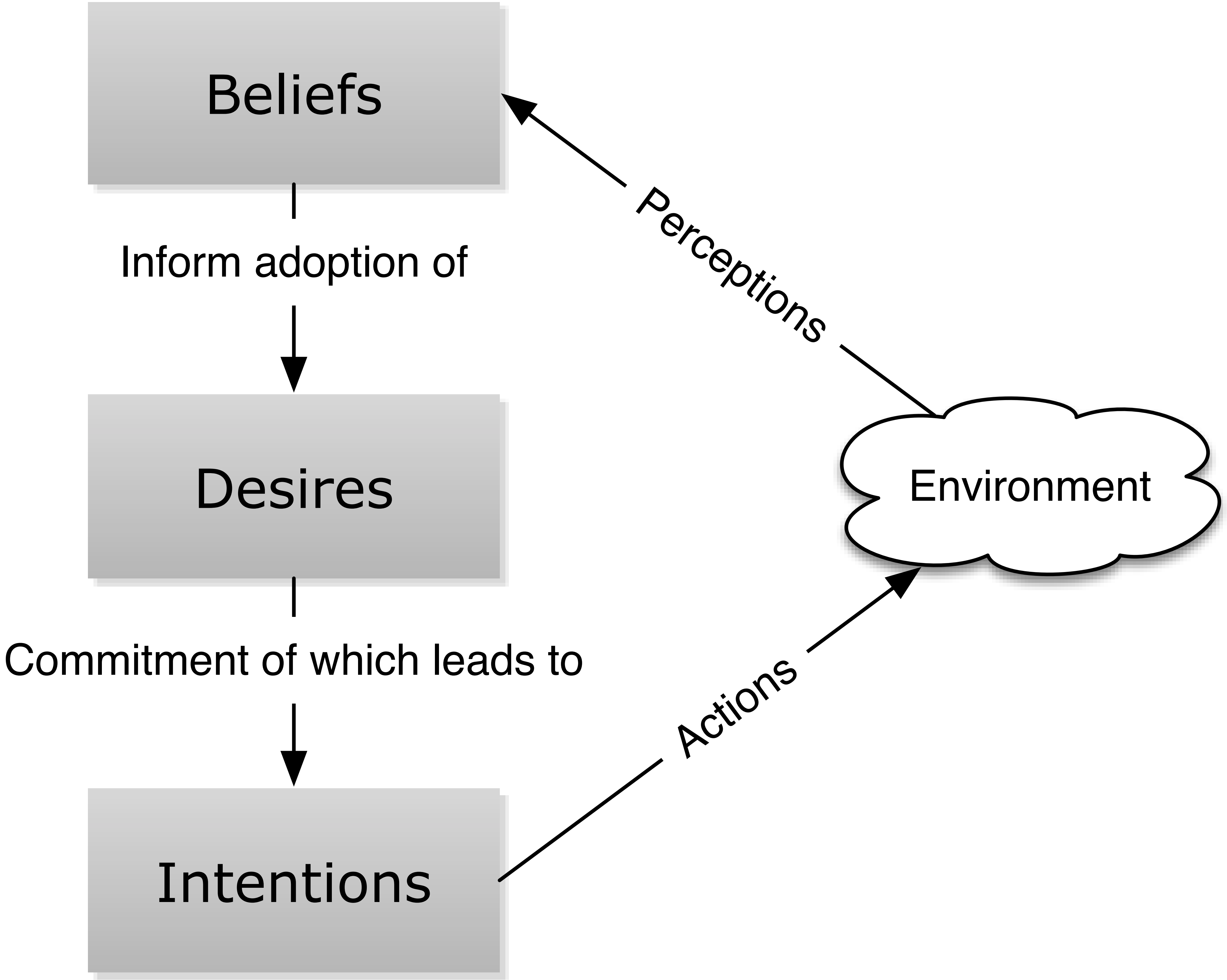
- A crash course in agents
- Agent programming abstractions
  - Declarative goals and planning
  - Normative Reasoning and Commitments
- An application in robotics

Agent programming

- Programming has progressed through:
  - machine code;
  - assembly language;
  - machine-independent programming languages;
  - sub-routines;
  - procedures & functions;
  - abstract data types;
  - objects;
- to **agents**.

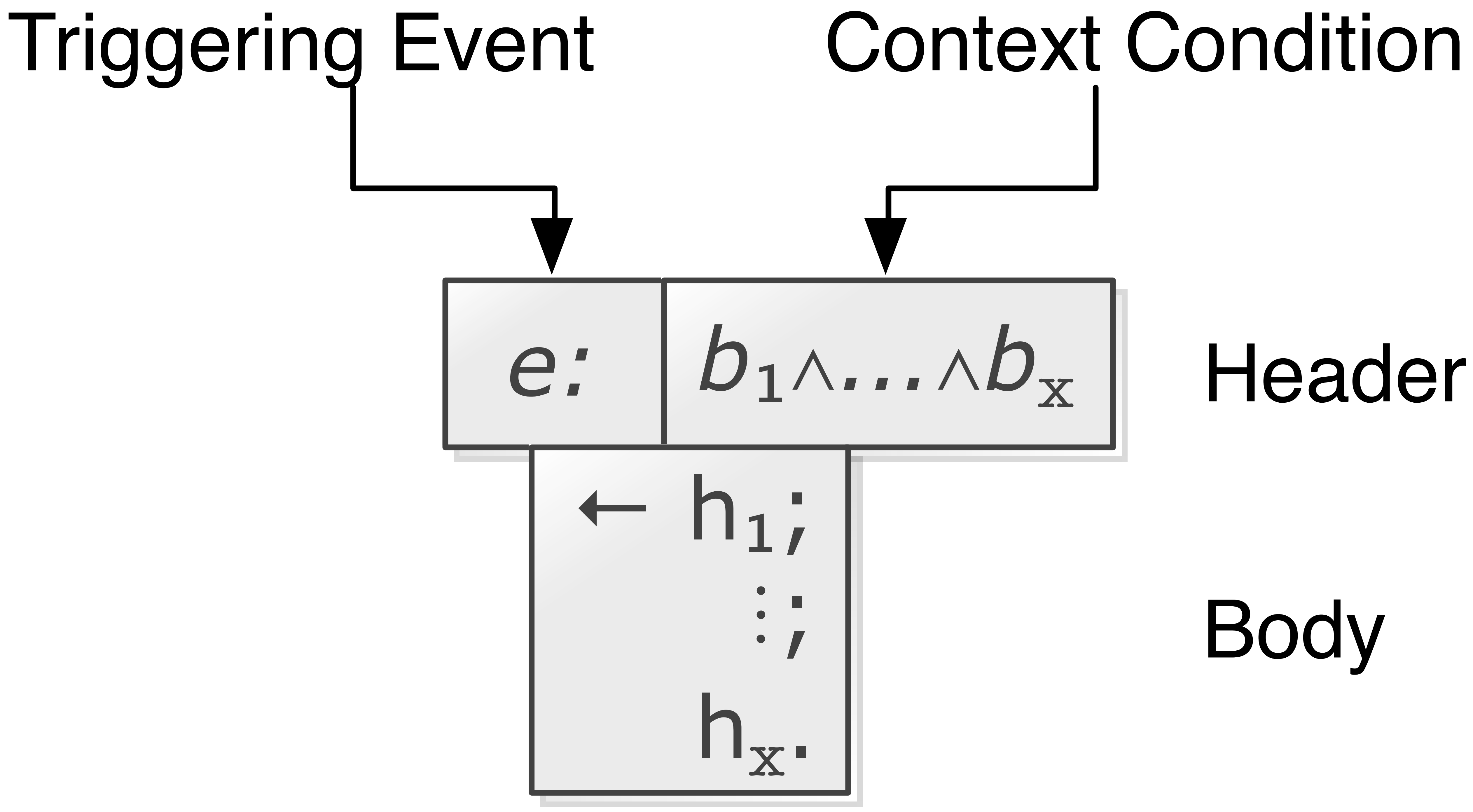
Overview of BDI Agents

- Philosophical model of human reasoning
- Based on three mental states:
    - Beliefs
    - Desires
    - Intentions

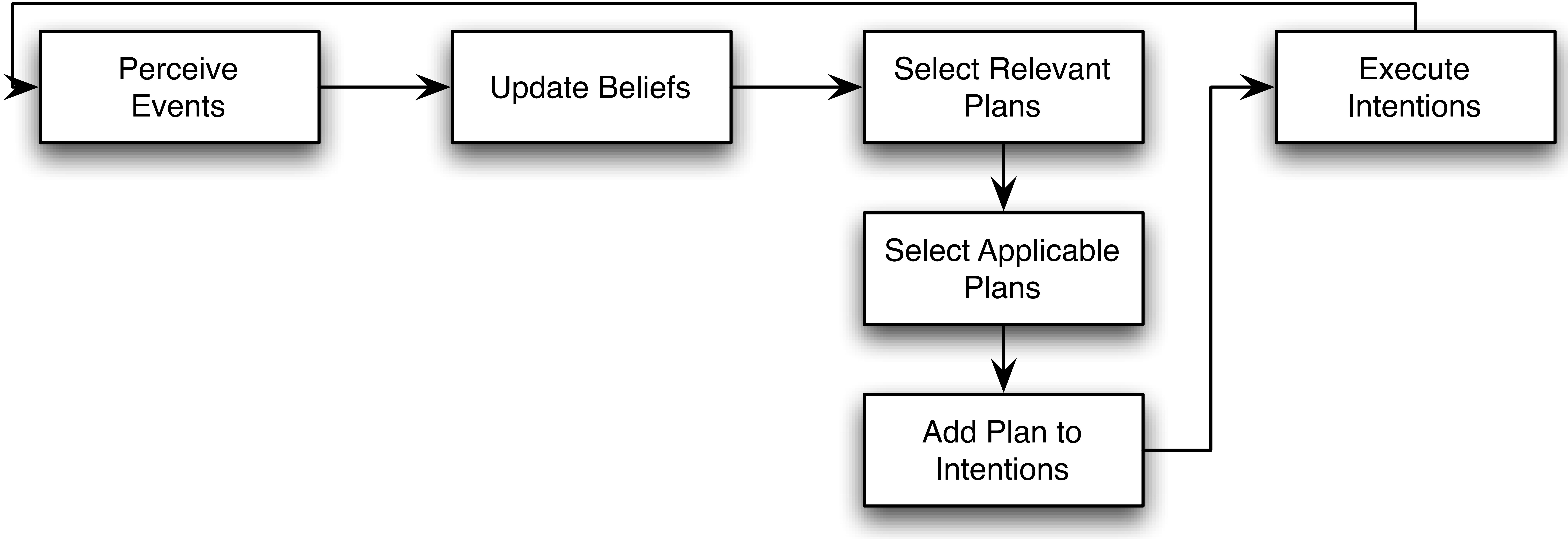


BDI Programming Languages

- Originated from the Procedural Reasoning System (PRS)
- Similar to Prolog in many respects
- Beliefs – First order logic atoms
- Plans – Procedures triggered by perception



BDI Reasoning



Plan Example

$$\left\langle +!goTo(C), hasVehicle(V), \begin{bmatrix} getVehicle(V), \\ moveTo(C) \end{bmatrix} \right\rangle$$

New event **Belief Base**

$!goTo(london)$ $hasVehicle(airplane)$

Resulting Plan

$$\begin{bmatrix} getVehicle(airplane), \\ moveTo(london) \end{bmatrix} (C = london, V = airplane)$$

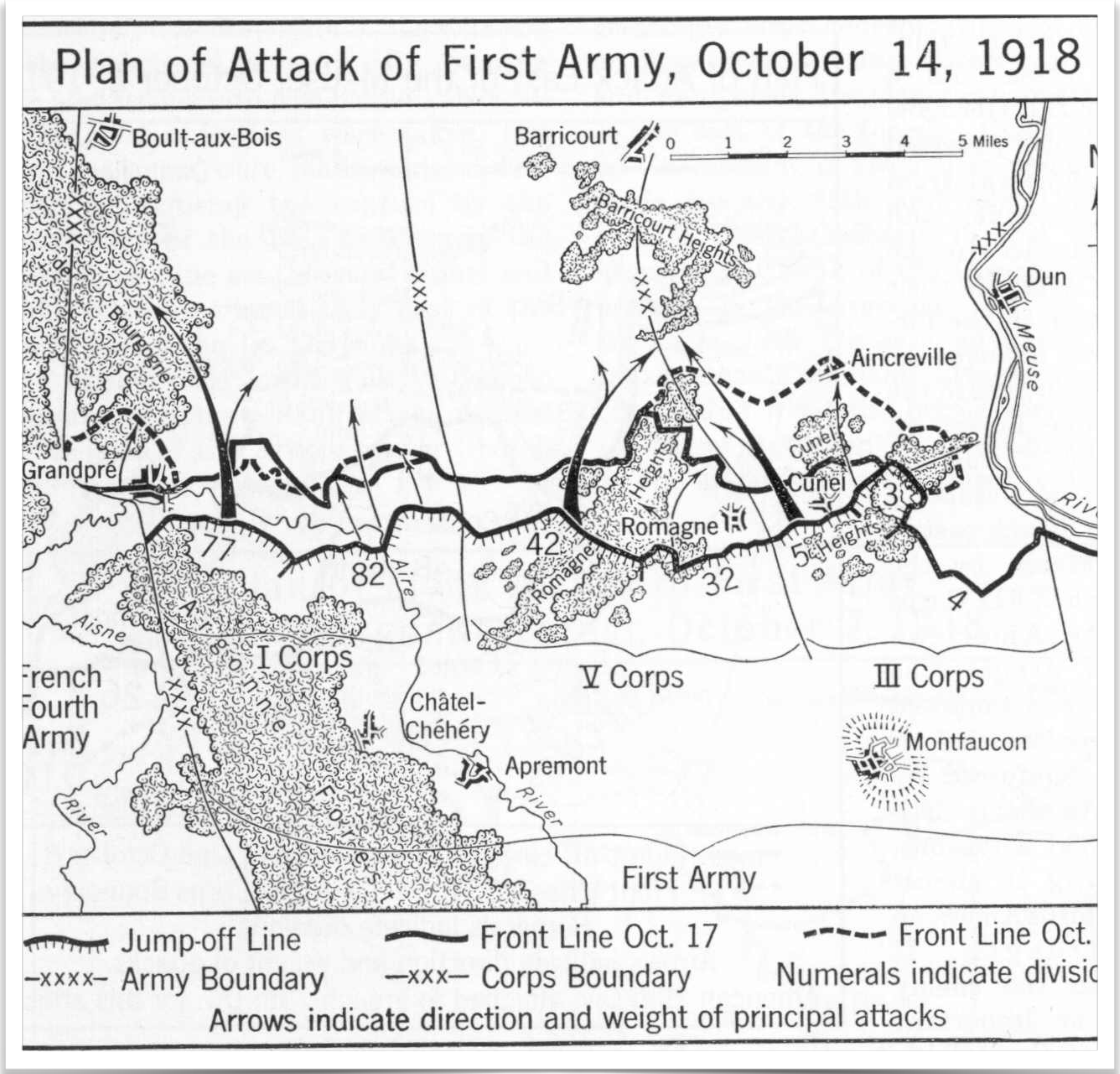
Goal Types

- Procedural Goals - Efficient, yet inflexible
  - Predefined encapsulated behaviours
  - Designer must foresee relevant plans
- Declarative Goals - expressive, but not trivial
  - Desired world states
  - Requires a more complex reasoning mechanism
- How to link desired world states to actions?

Planning in Agents

Automated Planning

- Necessary capability in autonomous systems
  - Deterministic (controlled environments)
  - Stochastic (real world)
- Applications
  - Plan recognition
  - Proactive assistance
  - Declarative Agent Programming



Planning in BDI agents

- Traditional agent languages rely on static plan libraries
- Introduction of first principles planning in the BDI programming languages to:
  - Expand agent’s capabilities at runtime
  - Support declarative goals
  - Comply with normative stipulations
  - Plan reuse algorithm based on the generation of context condition for newly created plans

Planning in AgentSpeak(L)

- Introduction of planning in the AgentSpeak(L) language
- Use of planner to support declarative goals
- Plan reuse algorithm based on the generation of context condition for newly created plans
- Expansion of the agent plan library

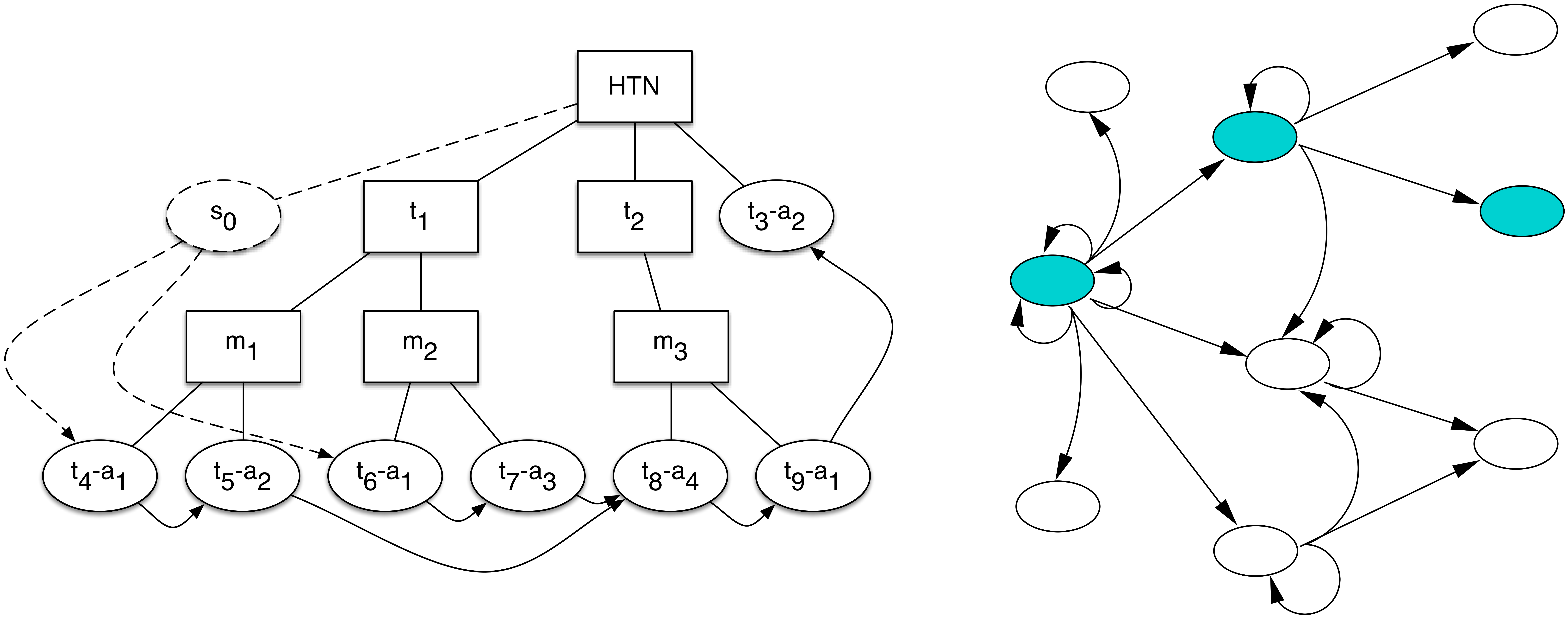
Planning Formalisms

- Planning is one of the main areas of AI research
- Research focuses on:
  - Relation between planning formalisms/algorithms to agent reasoning
  - Convertibility of deterministic planning formalisms to stochastic planning formalisms

Planning Formalism Translation

- Research focuses on utilizing more user-friendly formalisms
  - Target formalism: Markov Decision Process (MDP)
  - Base formalism: Hierarchical Task Networks (HTN)
- HTN-like abstractions are widely used in Agent Programming Languages (e.g. AgentSpeak)
- MDPs are a powerful mathematical model for probabilistic planning

Transformation



Probabilistic Planning

- Two approaches
  - Conversion of HTN to MDP
  - Planning through Earley Graph construction
- Goal
  - Convert a deterministic planning representation (with additional information) into a stochastic planning problem

Papers on Agent Planning

- MENEGUZZI, Felipe and DE SILVA, Lavindra. Planning in BDI Agents: **A survey of the integration of planning algorithms and agent reasoning**, In The Knowledge Engineering Review (KER), 2013.
- MENEGUZZI, Felipe and LUCK, Michael. **Declarative planning in procedural agent architectures**, In Expert Systems with Applications (ESWA), Vol. 40:16, 2013.
- MENEGUZZI, Felipe. **Motivations and Goal-Directed Autonomy**, in AAAI-10 Workshop on Goal-Directed Autonomy, 2010 (invited paper).

Normative Reasoning

Normative Systems

- Norms: mechanism to impose control on agent societies
- Define standards of acceptable behaviour
- Rely on explicit representations of:
  - Obligations/Prohibitions
  - Permissions
- Applications:
  - Electronic contracts
  - Simulated societies

Why Norms?

- Autonomous agents in heterogenous societies act to achieve individual goals
- Multiple agents acting simultaneously will interfere with each other (negatively)
- Strategies will be either:
  - One against everyone else (game theory)
  - One-to-one coordination (expensive)
  - Normative systems

Norms

- Represent desirable behaviours for members of a society
  - "Soft-constraints" on behaviour
  - General expectation of behaviour
  - **Rewards** for compliance + **Sanctions** for non-compliance
- Traditionally represented through conditional rules of the form:

$$\langle \nu, \alpha, \epsilon \rangle$$

- ν: Norm condition (Deontic Formula)
- α: Activation Condition
- ϵ: Expiration Condition

Deontic Logic

- Alethic modal logic deals with **what is** (or could be)
- Deontic logic deals with **what should be**
- Most common deontic modalities:
  - Obligations - Oq - it is obligatory that q
  - Permissions - Pq - it is permitted that q
    $Pq \leftrightarrow \neg O \neg q$
  - Prohibitions - Fq - it is not permitted that q
    $Fq \leftrightarrow O \neg q$

Deontic Logic

- This talk **is not** about deontic logic
  - A lot of work still being done in logic
- For our purposes we *greatly simplify* things in terms of:
  - States we **want** agents to achieve
  - States we **do not want** agents to achieve

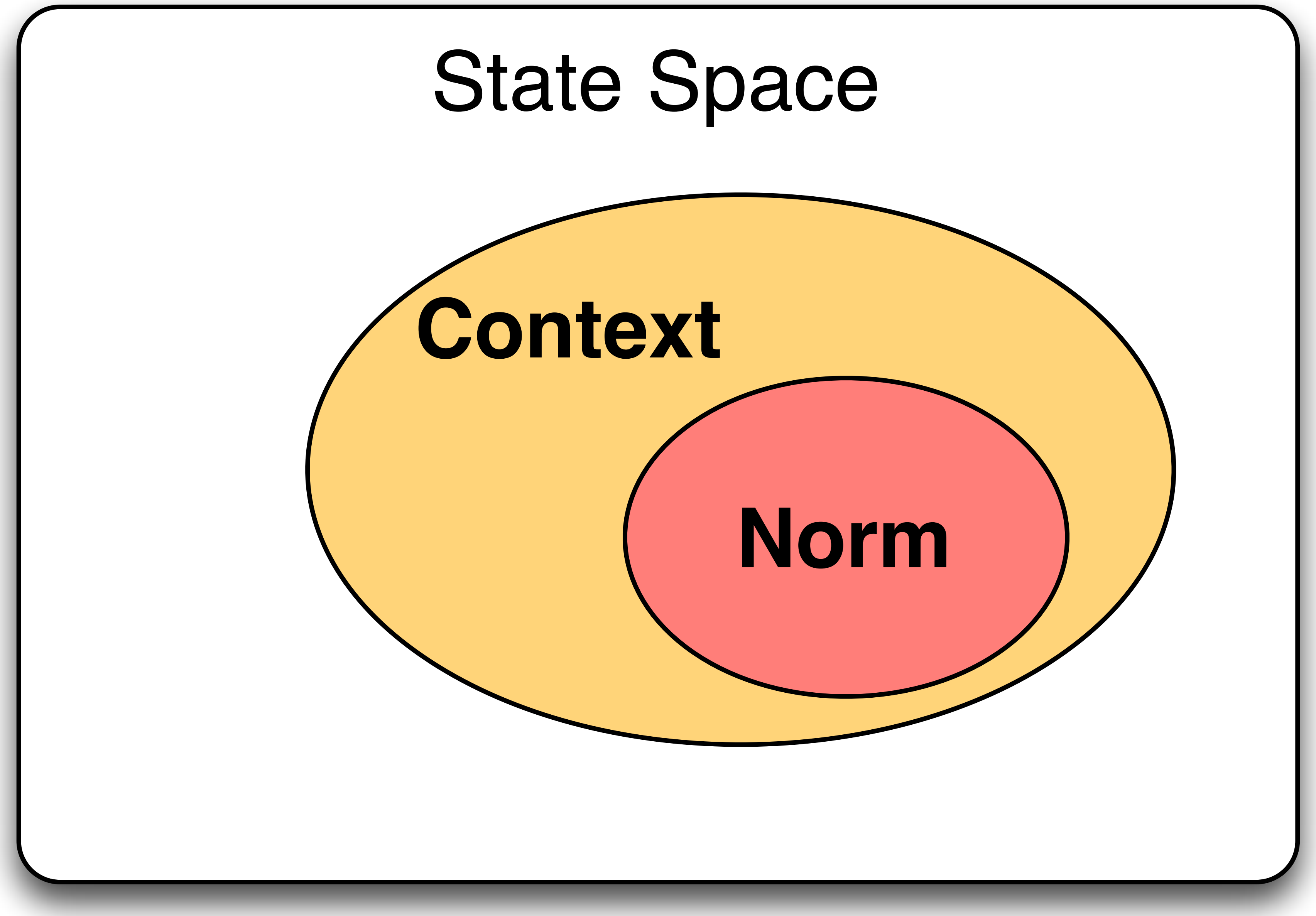
Traffic Light Example

$$\langle \mathbf{O}\, stop(A,P),$$
$$at(A,P) \wedge redlight(P),$$
$$\neg redlight(P)\rangle$$

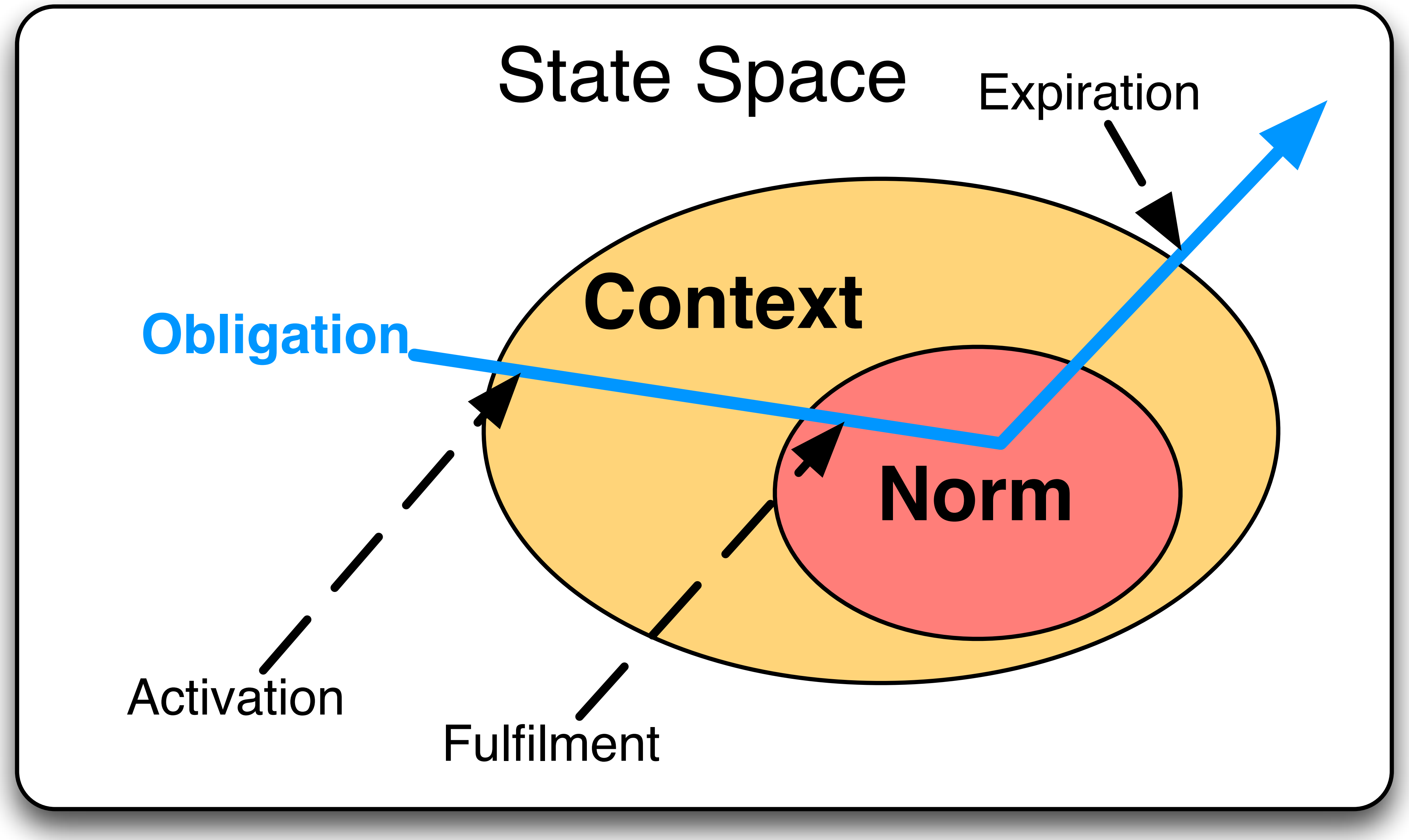
- Norm condition
- Activation condition
- Expiration condition

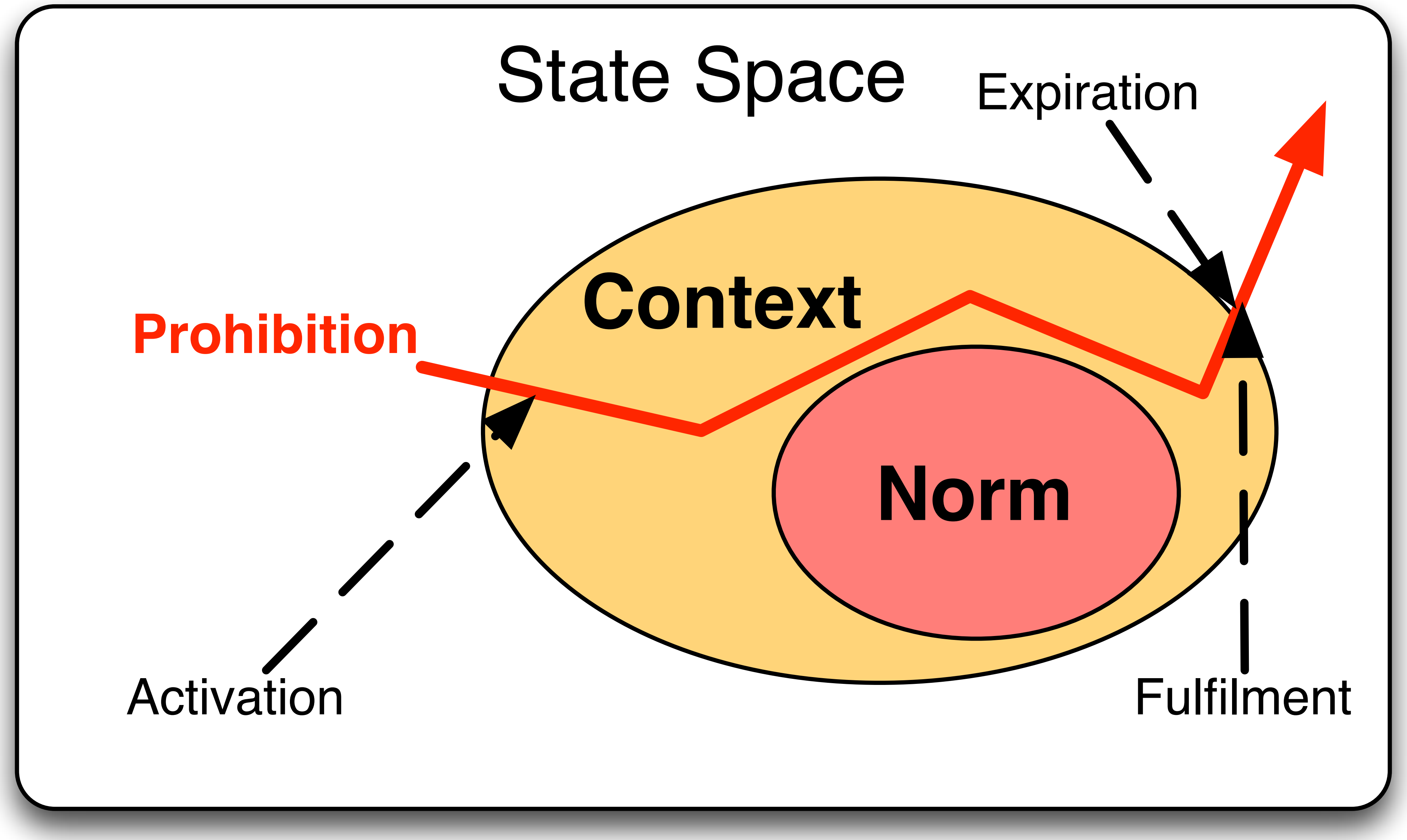
Norms and state-space

- Norm enforcement focuses on two sets of states
  - States between activation and expiration: **norm context**
  - States referred to by the **norm condition**
- Semantics of obligations sometimes differ

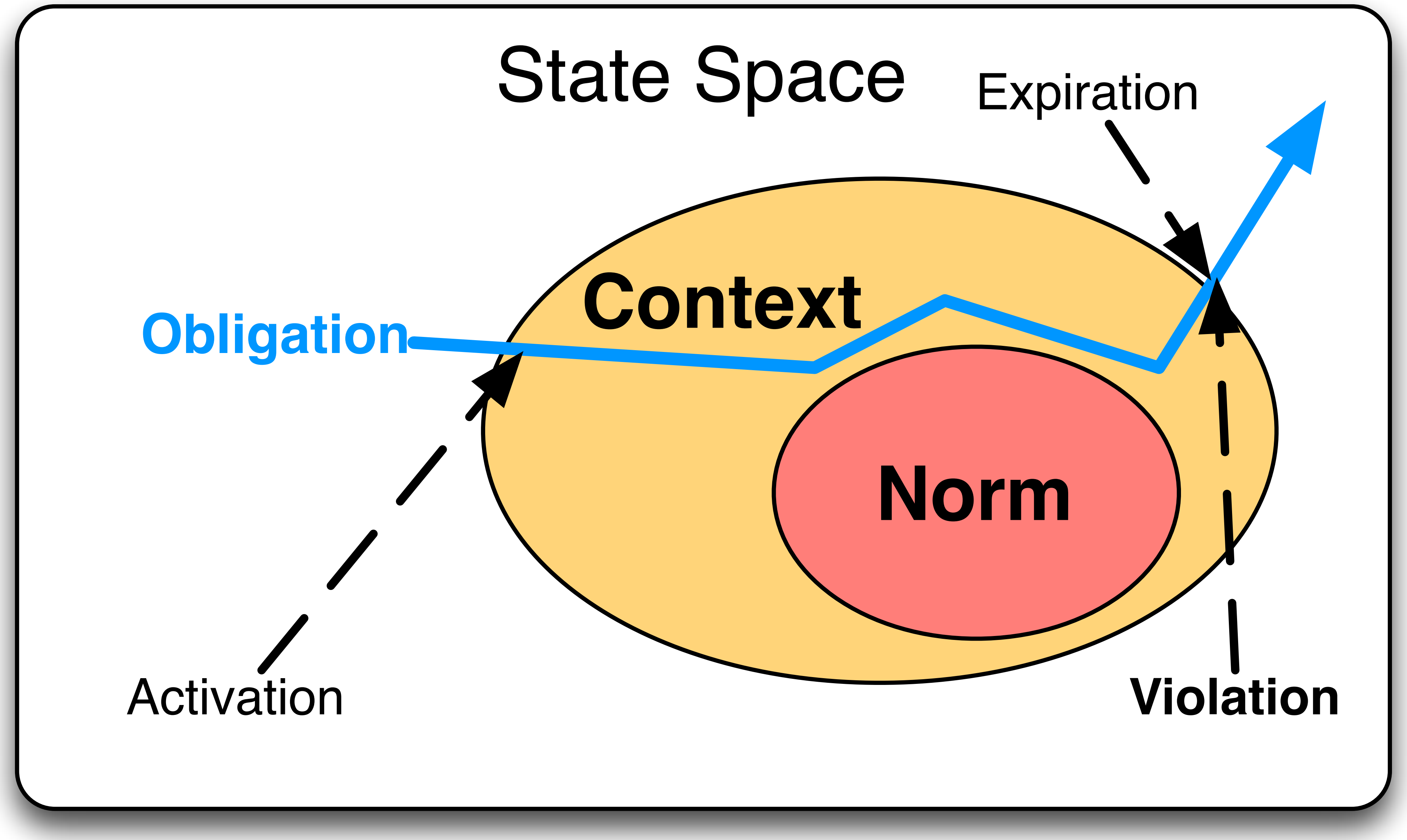


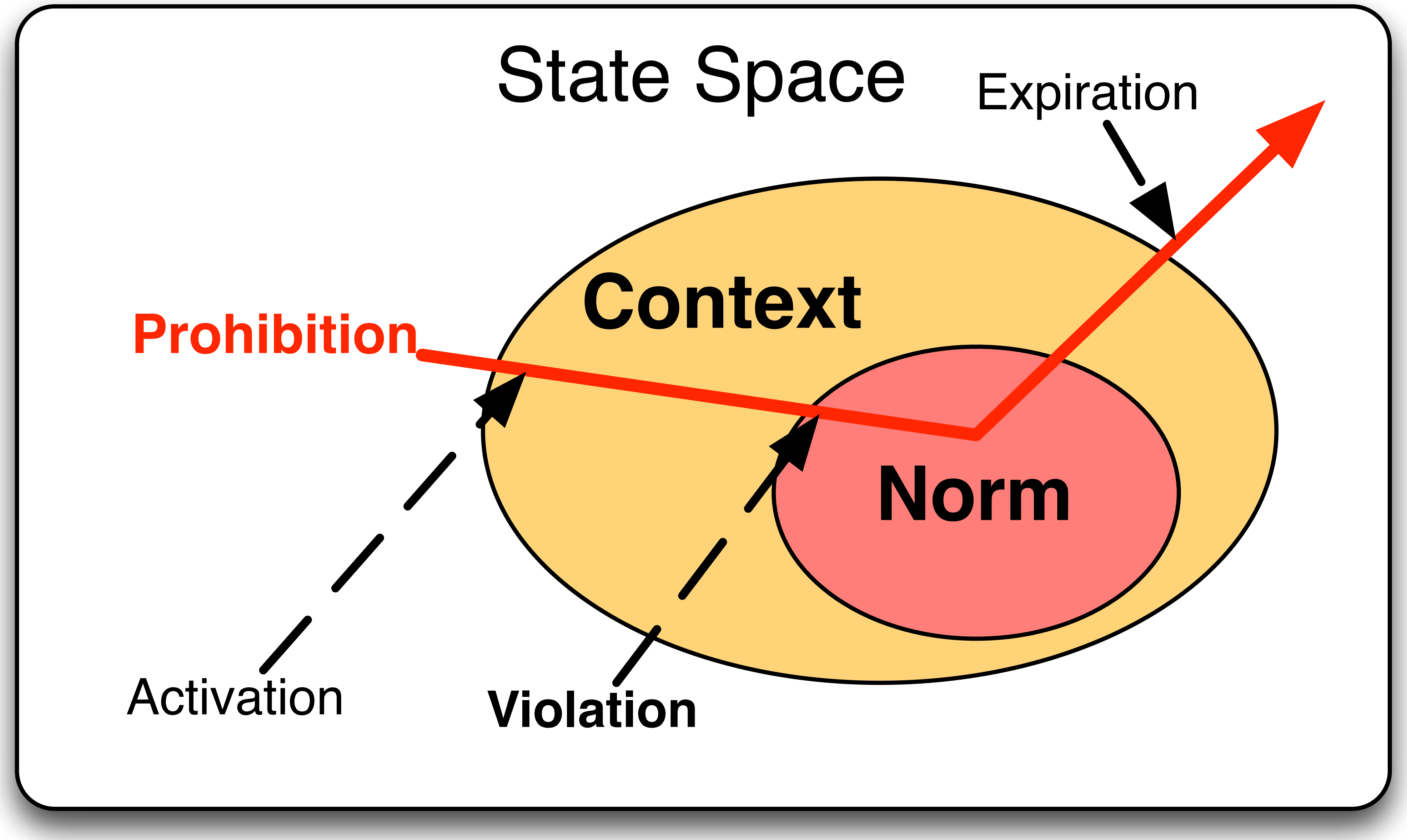
Norm Activation and Expiration





Norm Activation and Expiration





Practical Norm Reasoning

- Existing efforts largely focused
  - Logical aspects (deontic logic)
  - Macro-level (virtual organisations)
- Relatively few techniques for individual **agent behaviour**
  - Finite time/resources
  - Practical enforcement mechanisms

Practical Norm Reasoning

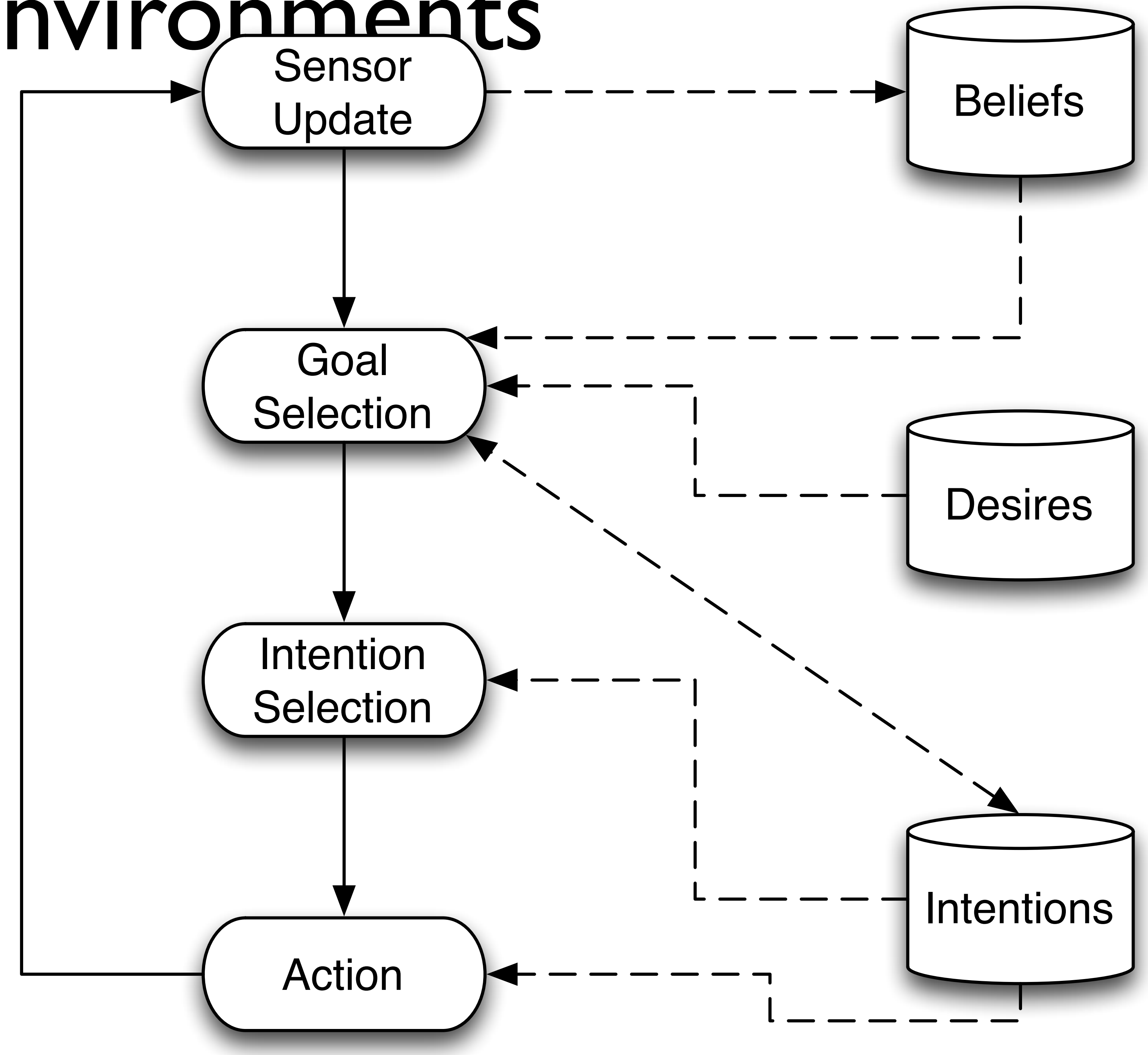
- How should an agent behave in a norm-driven society?
  - Norms as soft constraints
  - Dynamically changing sets of norms
  - Different enforcement mechanisms
  - Limited time/resources
- Depends on the assumptions on the environment

Environment Assumptions

- Deterministic/Stochastic
  - Plan selection
  - Decision theoretic planning
- Observable/Partially Observable
  - Norm inference / learning
- Explicitly multiagent
  - Reasoning about other agents/trust

Norms in the BDI model

- Assumption: deterministic, fully observable environments
- Reasoning within the BDI model
  - Beliefs - World model (from perception)
  - Desires - Overall objectives (from user)
  - Intentions - Committed objectives / plans (selected at runtime)
- Norms constrain intention selection



Norm representation

- Focuses on the operational aspect of norm compliance
- Norms are defined in the form
  - Normative Formula
  - Activation Condition
  - Expiration Condition
  - Id

$$\langle \nu, Act, Exp, id \rangle$$

Normative formula (ν)

- Annotated deontic formula is of the form
$$\mathsf{X}_{\alpha:\rho}\varphi \circ \Gamma$$
- Where X is the norm type:
  - O – for obligations
  - F – for prohibitions
- φ is the targeted formula (actions in a plan)
- And Γ is a conjunction of constraints

Previous Normative Systems

- Two extremes of norm processing
  - Blanket plan retractions (Normative AgentSpeak)
  - Every norm checked at every plan step (BOID)
- Decision about compliance too simplistic
  - Made before real repercussions are known or
  - Non-compliance simply not an option

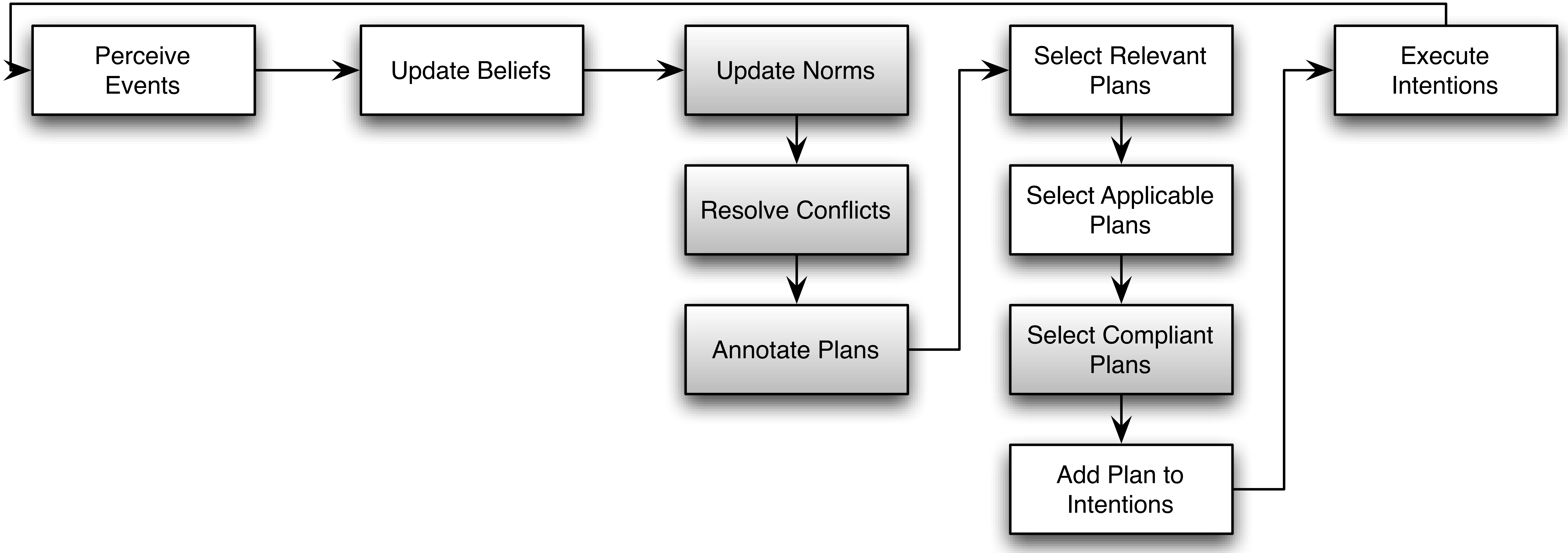
Architecture Desiderata

- We propose something in-between
  - Fine grained
  - Efficient
- Effect of norms calculated at norm receipt
- Decision to comply delayed as much as possible

Reasoning about Norms

- Three key processes:
  - Update norms (Resolve Conflicts)
  - Annotate Plan Library
  - Apply normative restrictions to plans

nu-BDI


Perceive Events
Update Beliefs
Update Norms
Resolve Conflicts
Annotate Plans
Select Relevant Plans
Select Applicable Plans
Select Compliant Plans
Add Plan to Intentions
Execute Intentions

Updating Norms

- Norms can be in two “states”
  - Abstract
  - Specific (or Active)
- When received by agent – abstract norms
- When activation condition holds – new specific norms created

Example Norm Update

- Abstract Norm

$$\left\langle \begin{array}{l} \mathsf{F}_{A:R} moveTo(C) \circ C = X, \\ tubeStrike(X), \\ \neg tubeStrike(X), \\ norm1 \end{array} \right\rangle$$

- New event occurs

$$tubeStrike(london)$$

- Specific Norm

$$\left\langle \begin{array}{l} \mathsf{F}_{A:R} moveTo(C) \circ C = london, \\ tubeStrike(london), \\ \neg tubeStrike(london), \\ norm1.1 \end{array} \right\rangle$$

- Specific Norm is deleted with event

$$\neg tubeStrike(london)$$

Annotating Plans

- Plans in the plan library are annotated as specific norms are created
- Normative formula is compared to steps in each plan
- Each step is associated with appropriate normative constraints

Example Plan Annotation

- Plan

$$\left\langle +!goTo(C), hasVehicle(V), \begin{bmatrix} getVehicle(V), \\ moveTo(C) \end{bmatrix} \right\rangle$$

- Specific Norm

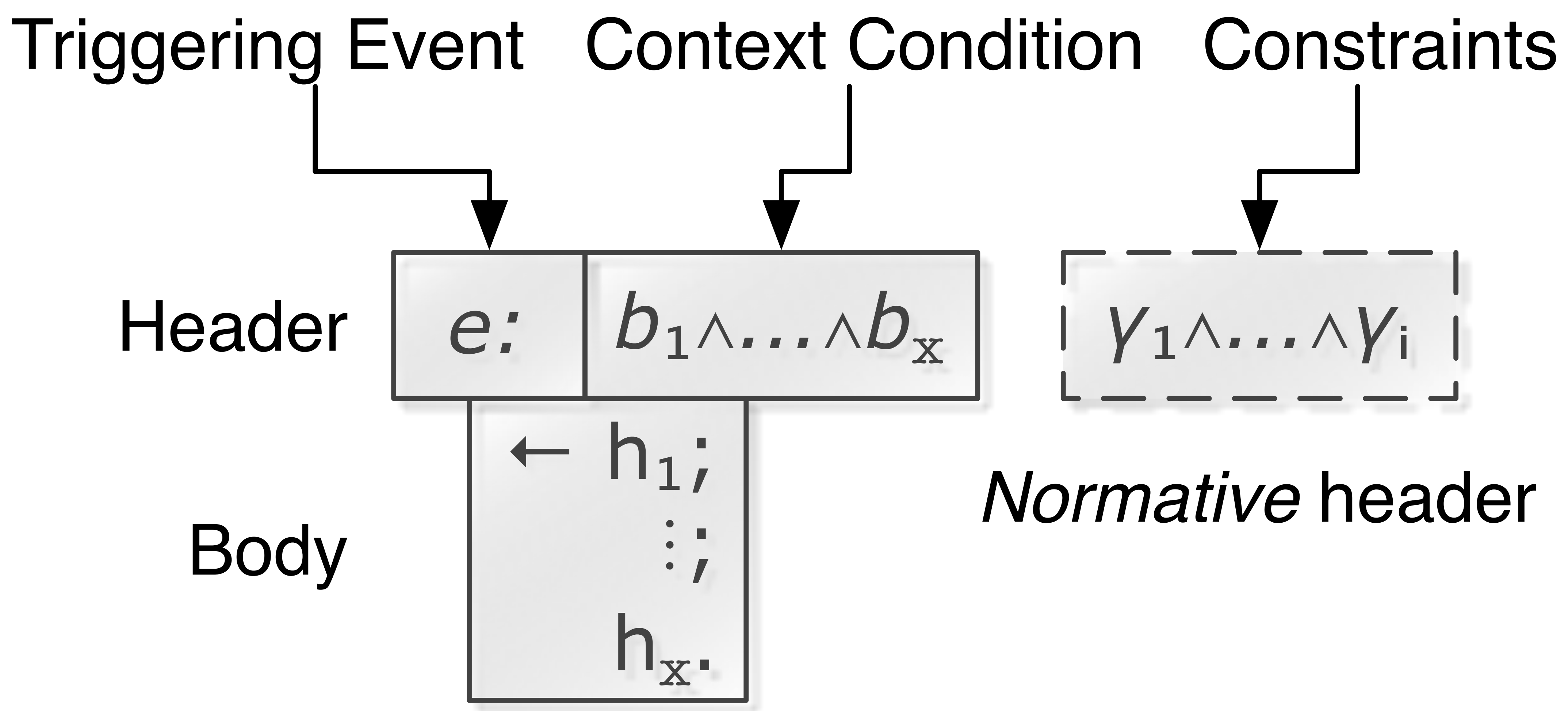
$$\left\langle \begin{array}{l} \mathsf{F}_{A:R} moveTo(C) \circ C = london, \\ tubeStrike(london), \\ \neg tubeStrike(london), \\ norm1.1 \end{array} \right\rangle$$

- Resulting annotated plan

$$\left\langle +!goTo(C), hasVehicle(V), \begin{bmatrix} getVehicle(V) \circ \mathsf{T}, \\ moveTo(C) \circ C \neq london \end{bmatrix} \right\rangle$$

Normative Plan Selection

- Similar to original plan selection
- Added check for satisfiability of a normative header
- Constraints from all steps



Example Plan Selection

$$\left\langle +!goTo(C), hasVehicle(V), \begin{bmatrix} getVehicle(V), \\ moveTo(C) \end{bmatrix} \circ C \neq london \right\rangle$$

New event **Belief Base**

$$!goTo(london) \quad hasVehicle(airplane)$$

Resulting Plan

$$\begin{bmatrix} getVehicle(airplane), \\ moveTo(london) \end{bmatrix} (C = london, V = airplane)$$

But

$$(C = london \wedge C \neq london) \rightarrow \perp$$

Papers on Normative Reasoning

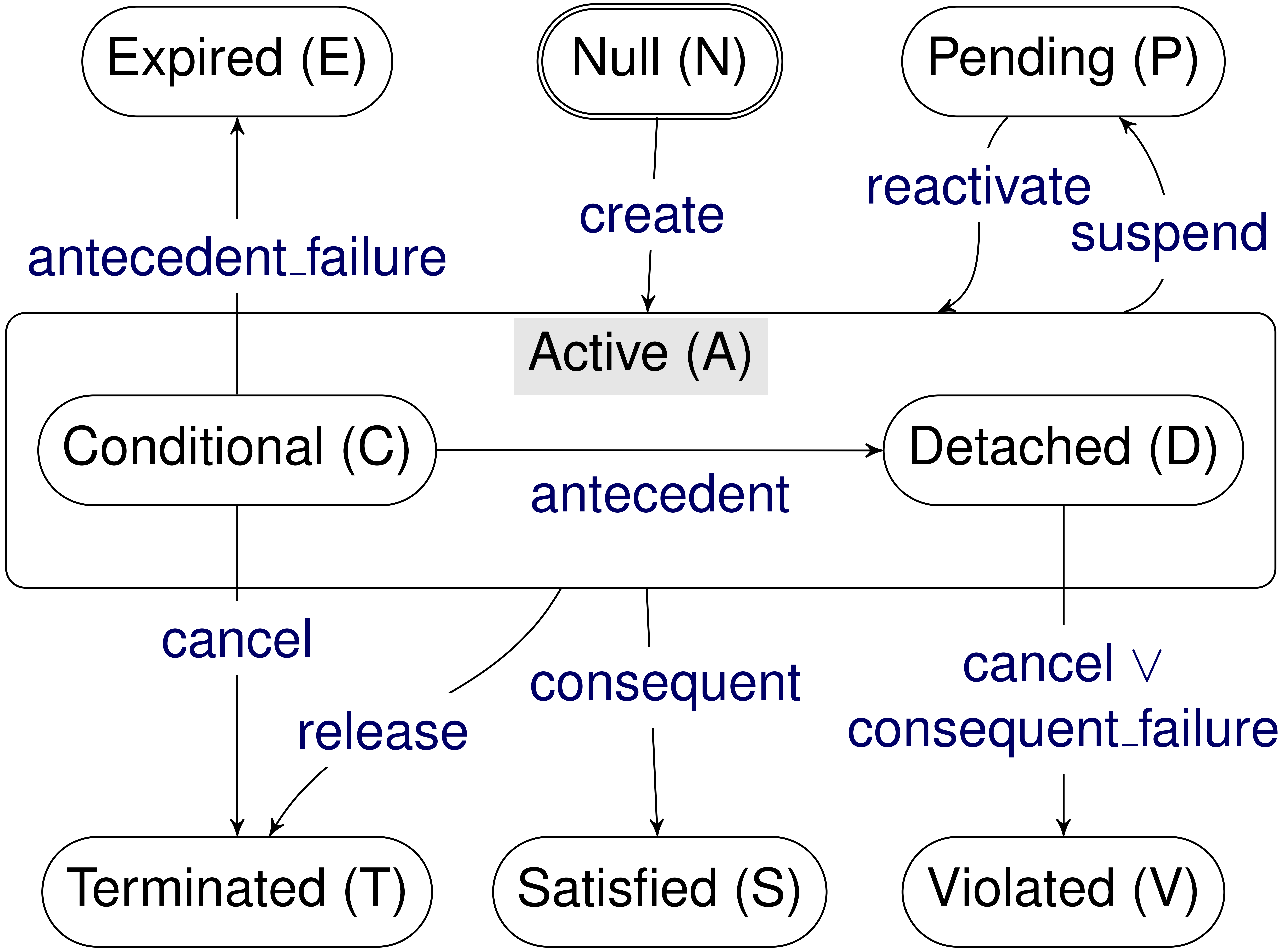
- FAGUNDES, Moser; OSSOWSKI, Sascha; and MENEGUZZI, Felipe. **Imperfect norm enforcement in stochastic environments: an analysis of efficiency and cost tradeoffs**, In Proc. 14th IBERAMIA, 2014.
- ALRAWAGFEH, Wagdi and MENEGUZZI, Felipe. **Utilizing Permission Norms in BDI Practical Normative Reasoning**, In COIN 2014 @AAMAS, 2014.
- Meneguzzi and Luck. **Norm-based behaviour modification in BDI agents**. AAMAS (1) 2009: 177-184.
- Meneguzzi et al. **Nu-BDI: Norm-aware BDI Agents**. EUMAS 2012

Goals and Commitments

Motivation

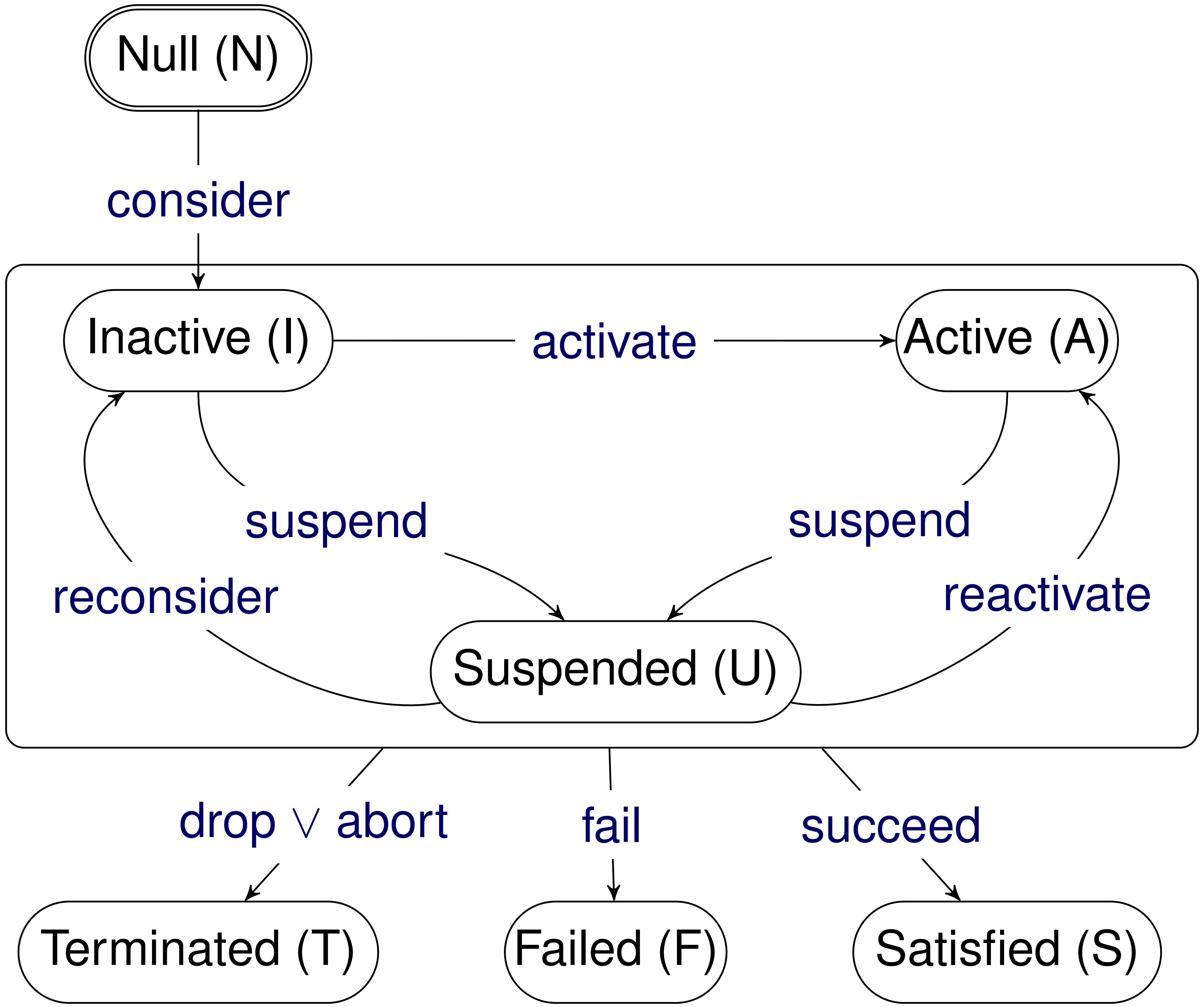
- Commitments have been extensively studied in MAS
  - Encode high-level social relations between agents
  - Define communication protocols among agents (business processes)
- Previous formalizations
  - Operational semantics for goals and commitments, and their interaction
  - Propositional planning formalization

Commitment Lifecycle



- Formally C(Debtor, Creditor, antecedent, consequent)
- E.g. C(buyer,seller,goods,paid)

Goal Lifecycle



- Formally G(Agent, pg, s, f)
- E.g. G(buyer, needsgoods, goods,deadline)

Relating Commitments and Goals

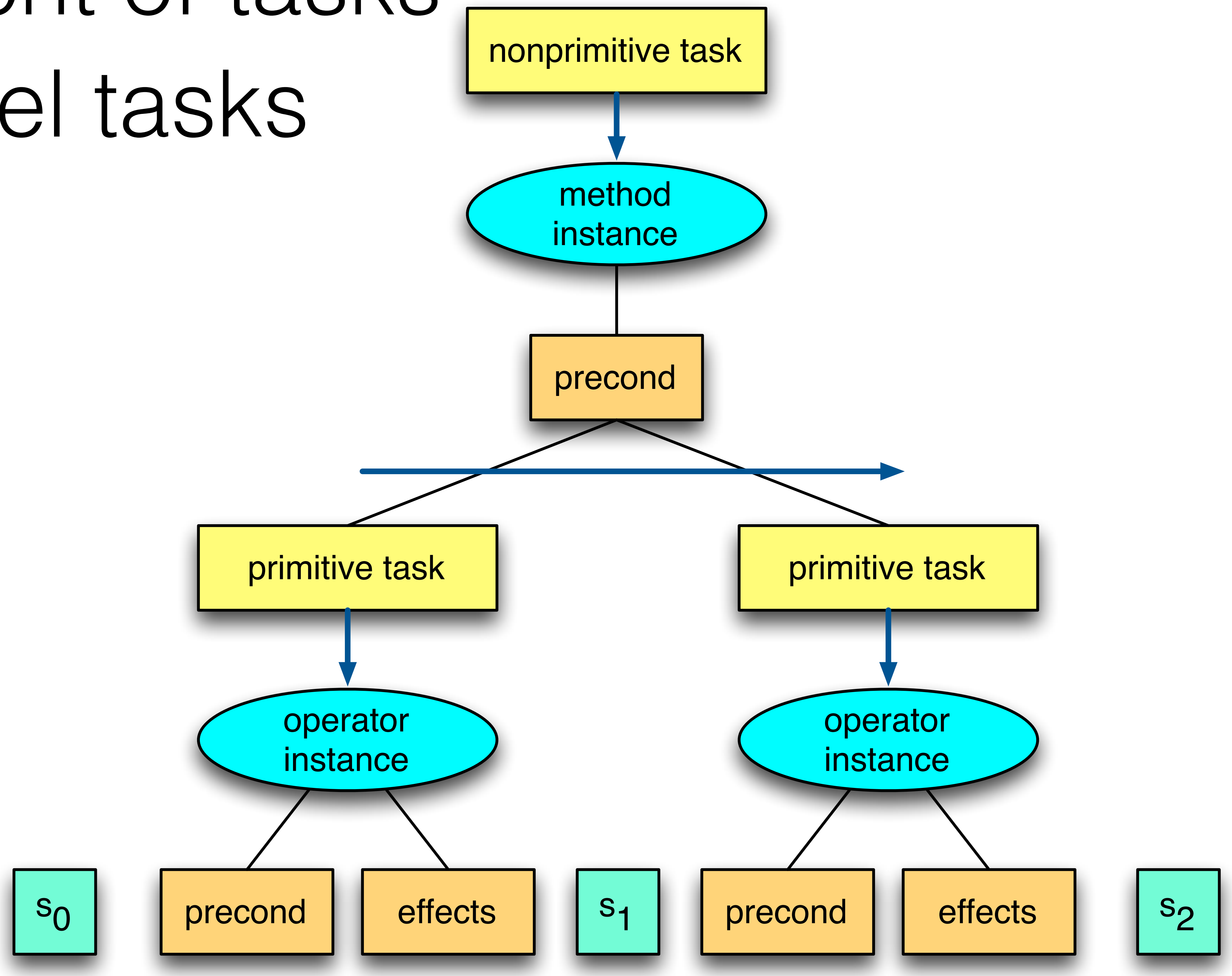
- Practical Rules relating commitments and goals
  - Let G = G(buyer, $\top$, goods, $\perp$)
    and C = C (buyer, seller, goods, pay)
  - **Entice Rule**: If G is active and C is null, buyer creates C

$$\frac{\langle G^A, C^N \rangle}{create(C)}$$

  - Motivation: *Buyer* can achieve its goals of goods by creating the commitment to pay for them to *Seller*

Hierarchical Task Network Planning

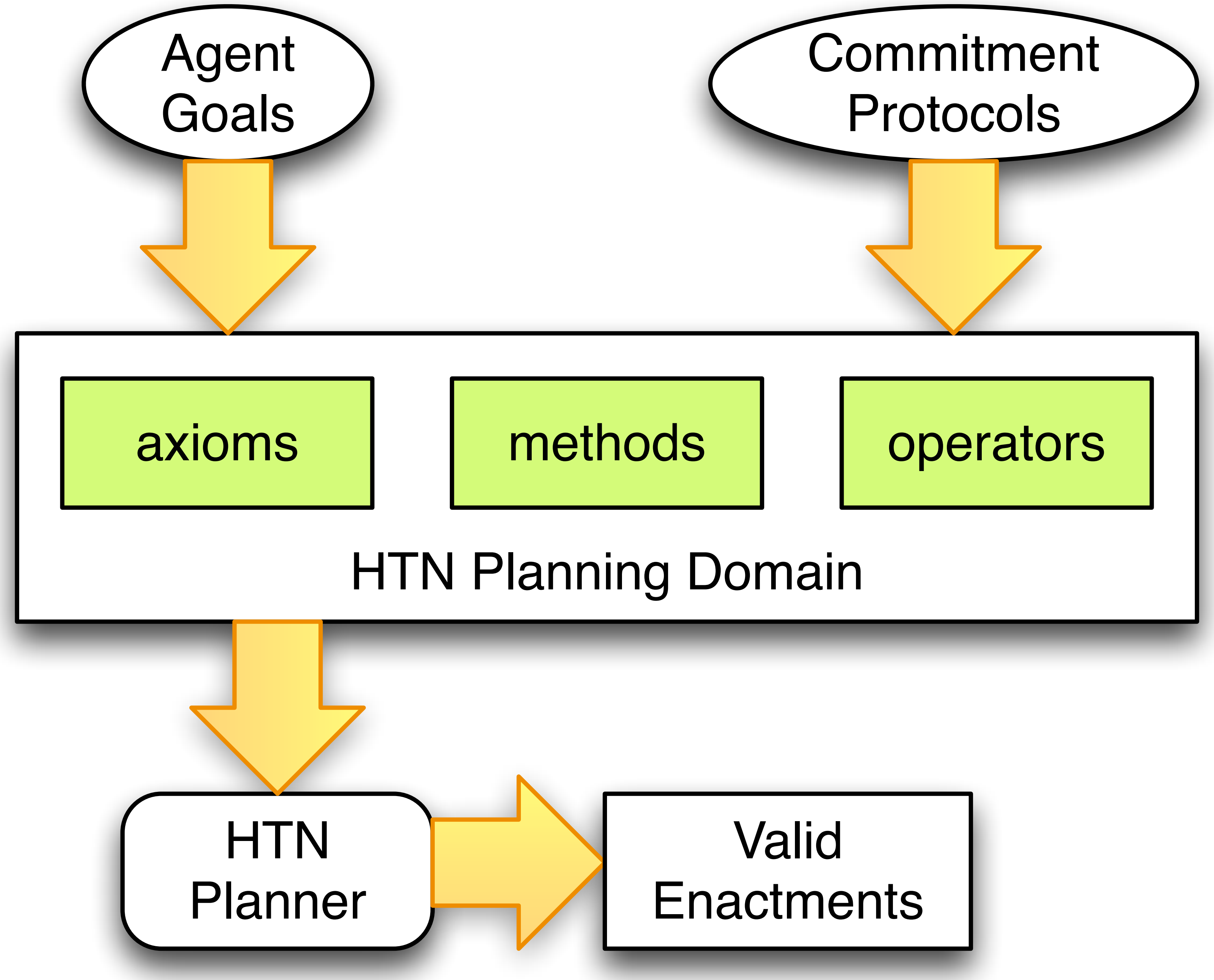
- Generates a plan by successive refinement of tasks
  - Non-primitive Tasks - abstract, high-level tasks to be decomposed
  - Primitive Tasks - cannot be further decomposed (operators)
- Multiple implementations (e.g. JSHOP2, SHOP2)



- Abstraction of choice for agent programming languages

HTN Planning for Commitments and Goals

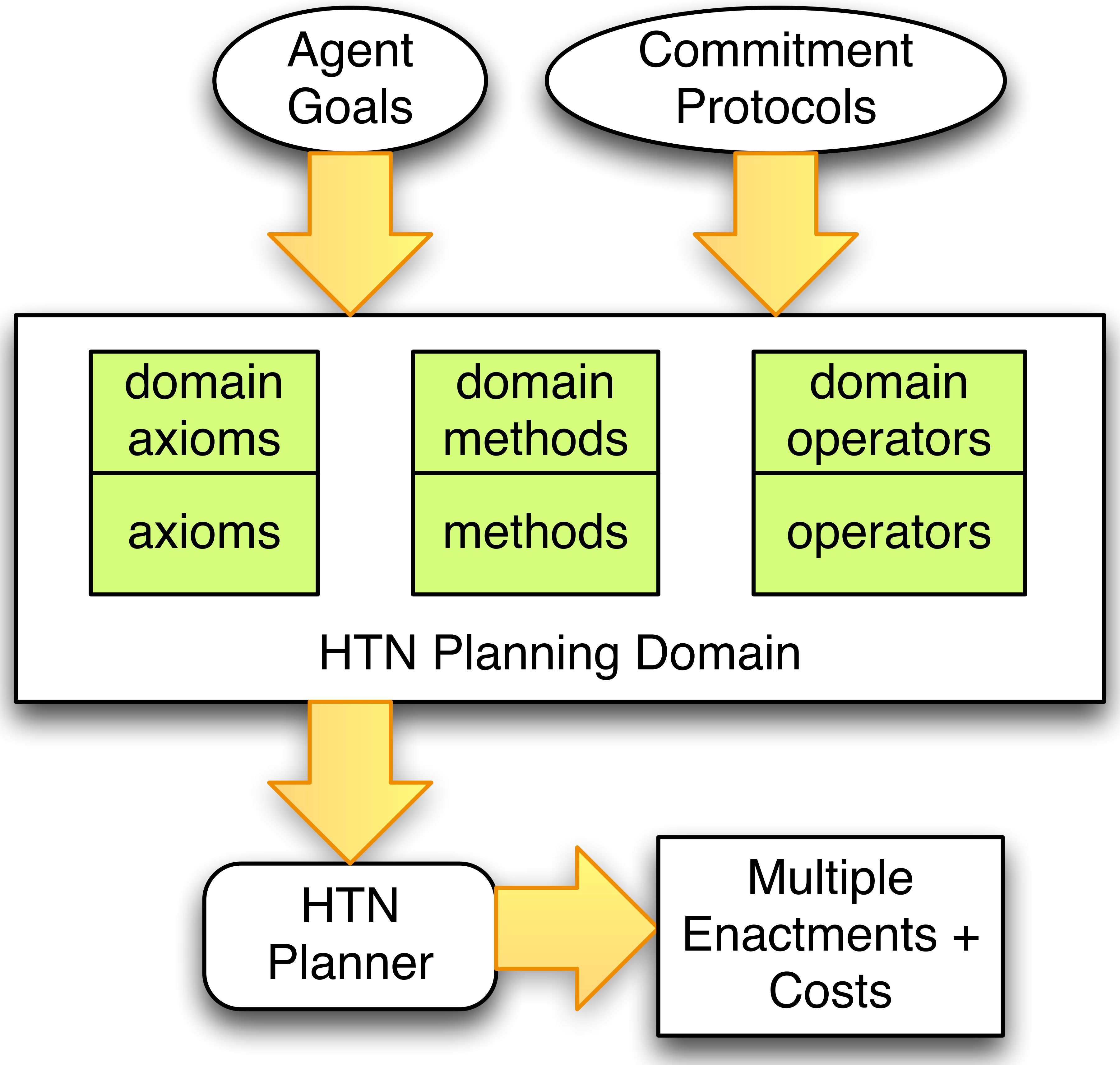
- Formalization of commitment protocols in terms of HTN planning
    - Axioms enforcing state transition model for goals and commitments
    - Planning Operators describing transitions (e.g. create, suspend, etc.)
    - HTN Methods for practical rules (e.g. entice, negotiate, etc.)



- Allows HTN planner to be used to validate commitment protocols

A first-order formalization

- Propositional formalization had several limitations
  - Limited expressivity
- New First-order formalization:
  - Domain independent axioms, methods and operators
  - Domain dependent axioms, costs, methods and operators
  - Useful patterns of behavior



Domain Independent Axioms & Operators

Commitment Axioms

$$null(C, Ct, \vec{Cv}) \leftarrow \neg var(C, Ct, \vec{Cv})$$
$$conditional(C, Ct, \vec{Cv}) \leftarrow active(C, Ct, \vec{Cv}) \wedge \neg p(C, Ct, \vec{Cv})$$
$$detached(C, Ct, \vec{Cv}) \leftarrow active(C, Ct, \vec{Cv}) \wedge p(C, Ct, \vec{Cv})$$

Goal Axioms

$$null(G, Gt, \vec{Gv}) \leftarrow \neg var(G, Gt, \vec{Gv})$$
$$\begin{aligned} inactiveG(G, Gt, \vec{Gv}) \leftarrow & \neg null(G, Gt, \vec{Gv}) \\ & \wedge \neg f(G, Gt, \vec{Gv}) \wedge \neg s(G, Gt, \vec{Gv}) \\ & \wedge \neg terminalG(G, Gt, \vec{Gv}) \wedge \neg suspendedG(G, Gt, \vec{Gv}) \\ & \wedge \neg activeG(G, Gt, \vec{Gv}) \end{aligned}$$

Commitment Operators

$$\begin{aligned} \langle & \textbf{operator } !create(C, Ct, De, Cr, \vec{Cv}), \\ & \textbf{pre}(commitment(C, Ct, De, Cr) \wedge null(C, Ct, \vec{Cv})), \\ & \textbf{del}(), \textbf{add}(var(C, Ct, \vec{Cv}))\rangle \end{aligned}$$
$$\begin{aligned} \langle & \textbf{operator } !suspend(C, Ct, De, Cr, \vec{Cv}), \\ & \textbf{pre}(commitment(C, Ct, De, Cr) \wedge active(C, Ct, \vec{Cv})), \\ & \textbf{del}(), \textbf{add}(pending(C, Ct, \vec{Cv}))\rangle \end{aligned}$$

Goal Operators

$$\begin{aligned} \langle & \textbf{operator } !consider(G, Gt, X, \vec{Gv}), \\ & \textbf{pre}(goal(G, Gt, X) \wedge null(G, Gt, \vec{Gv}) \wedge pg(G, Gt, \vec{Gv})), \\ & \textbf{del}(), \textbf{add}(var(G, Gt, \vec{Gv}))\rangle \end{aligned}$$
$$\begin{aligned} \langle & \textbf{operator } !activate(G, Gt, X, \vec{Gv}), \\ & \textbf{pre}(goal(G, Gt, X) \wedge inactiveG(G, Gt, \vec{Gv})), \\ & \textbf{del}(), \textbf{add}(activatedG(G, Gt, \vec{Gv}))\rangle \end{aligned}$$

Domain Dependent Definitions

- Axioms plus Domain-dependent operators
- Commitment Axioms

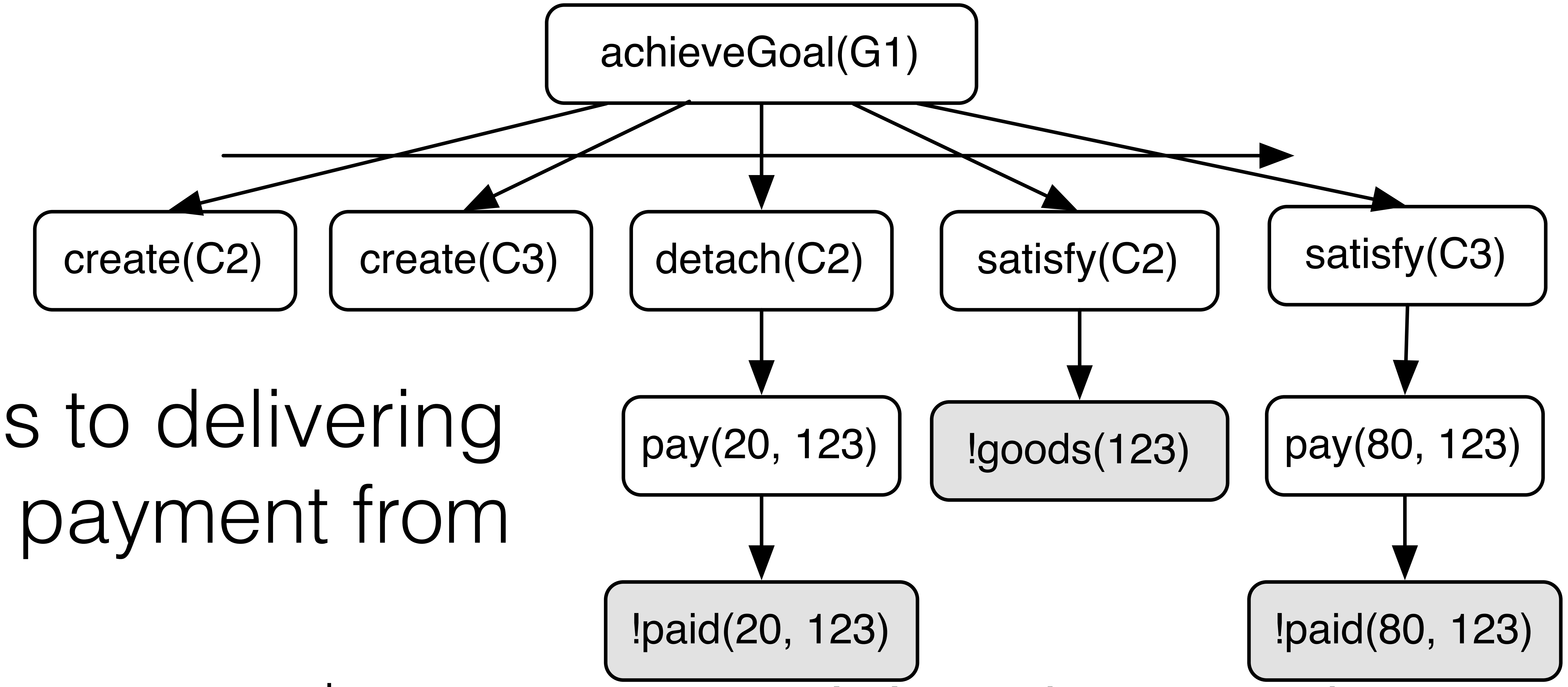
$$p(C, Ct, \vec{Cv}) \leftarrow commitment(C, Ct, De, Cr) \wedge \varphi$$
$$q(C, Ct, \vec{Cv}) \leftarrow commitment(C, Ct, De, Cr) \wedge \varkappa$$

- Goal Axioms

$$pg(G, Gt, \vec{Gv}) \leftarrow goal(G, Gt, X) \wedge \varpi$$
$$s(G, Gt, \vec{Gv}) \leftarrow goal(G, Gt, X) \wedge \varsigma$$
$$f(G, Gt, \vec{Gv}) \leftarrow goal(G, Gt, X) \wedge \vartheta$$

- Axioms Generated automatically using a compilation tool
- Plus any domain-specific operators (e.g. purchase, ship, etc)

Patterns of Behavior



- Concession Pattern
  2 commitments
  - C2 - merchant commits to delivering the goods upon a $20 payment from the customer
  - C3 - customer commits to pay $80 upon receiving the goods
- By creating commitments C2 and C3, the customer has one possible way of achieving its goal

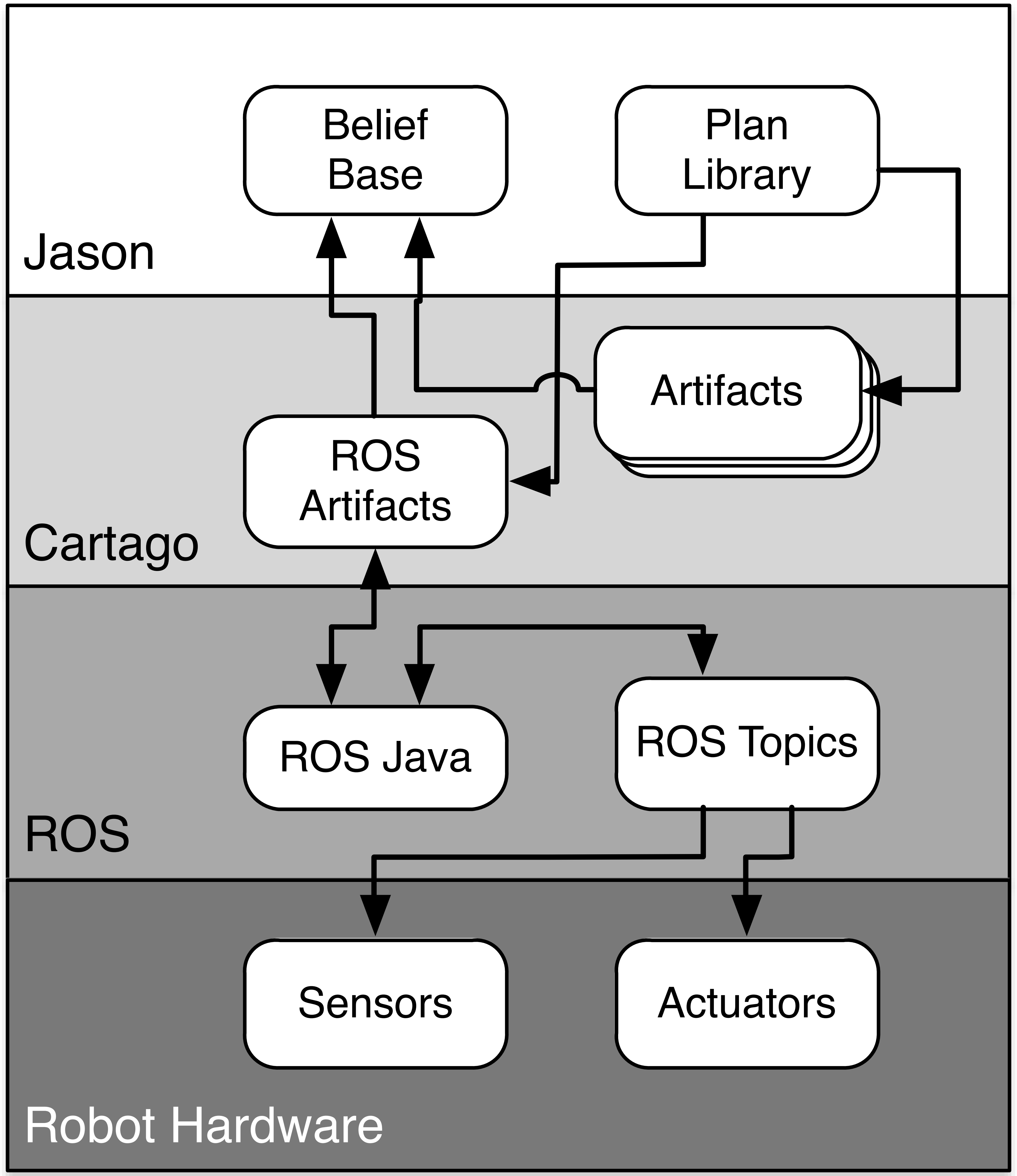
Papers on Goals and Commitments

- MENEGUZZI, Felipe; TELANG, Pankaj and SINGH, Munindar P. **A First-Order Formalization of Commitments and Goals**, In Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI), Bellevue, WA, USA, 2013.
- TELANG, Pankaj; MENEGUZZI, Felipe and SINGH, Munindar P. **Hierarchical Planning about Goals and Commitments**, In Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Saint Paul, MN, USA, 2013.

An Application in Robotics

JaCaROS

- Introduction of robotics abstractions to the AgentSpeak(L) language
    - Implementation in JaCa (Jason+Cartago)
    - Low-level robotic control using ROS
    - Uses artifact abstraction for robotic devices
- Implemented in simulation (Gazebo) and in physical robot (Turtlebot)



Robot Videos

Questions?