Distributed Planning in Multi-agent Environment by Means of Social Commitments

Minimal Ph.D. Proposal

Antonín Komenda

Czech Technical University in Prague
Gerstner Laboratory, Agent Technology Center

komenda@agents.felk.cvut.cz

August, 2009

Contents

Chapter 1

Introduction

Planning in a multi-agent environment is considerably complex especially due to its dynamism, non-determinism, and the distributed nature of the agents. This properties lead us to a requirement of distributed plans as robust as possible. For such environment, the usage of classical planning methods is not sufficient or even not possible. If the agents have to accomplish their mutual goals in the most effective way, it is necessary to ensure that their actions are not contradictory and that their interactions support the overall cooperation of the community. The classical methods of agents' behavior are usually based on the deliberation of their neighborhood and knowledge. Actions of an agent are the result of such a process and they can affect the environment in the intended sense. With the help of a robust planning, the agents gain an ability to cogitate an evolution of the environment as their plans describe their future. If an agent knows its anticipated future, it can utilize this knowledge both for itself or share it with other agents to improve their mutual cooperation.

The main objective of the dissertation research project, introduced in this proposal, is to define a unified formalization of the problem, to create a plan execution model, and to design and implement planning algorithms capable of handling the dynamism, non-determinism, and the distributed nature of the environment. The research results will be validated in the multi-agent environments motivated by real-world-based scenarios. The domains of the scenarios are for example (i) complex disaster relief [17], (ii) logistics with environment uncertainty [15], and (iii) distributed UAV surveillance missions [18].

The organization of this document is as follows. The remainder of this chapter introduces the research objectives, and approach to the solution. In Section 2, it is presented the state of the art summary of the three related topics of planning, agent plan execution models, and agents' commitments. Particular student's contributions to the related research fields are summarized in Section 3. The work is concluded by the dissertation plan.

1.1 Research Objectives

As it was already mentioned, the target environment is *dynamic*, *non-deterministic*, and naturally *distributed*. The dynamism is caused by the behaving agents in the environment and additionally, also by an environment model (e.g. simulation of the physical world, asynchronous inter-agent communication, self-interested goals of the agents). Since the agents are distributed entities in the environment, the environment is naturally distributed and non-deterministic from the perspective of the planning problem.

There are four objectives of the dissertation research project:

1. **Planning problem representation** – The first objective is to formalize a representation of the planning problem. The formalization will be used for description of the planning domains, the planning principles, and the execution model of the plans (see objective 2). The planner needs an unified formal description of both the environment, and the capabilities of the agents (objective 3).

2. **Plan execution model** – The plan has to be appropriately executed in the environment. Thus the second objective is to significantly improve the execution model to be capable of planning in the dynamic non-deterministic distributed environment.

3. **Planner** – The main objective is to design a planner solving the problem defined in the representation based on results of the objective 1. The plans have to be executable by the agents, according to the plan execution model introduced in objective 2. The key features of the planner ought to be the robustness of the plans, stability of the execution and efficient adjustability of the planning process.

4. **Experimental validation in a multi-agent prototype** – The last objective is to validate the effectiveness and efficiency of the approach in the multi-agent environments motivated by real-world-based scenarios and in synthetic experiments. The evaluation metrics will be based on the key features of the plans (resulting from objective 3). The successfulness will be measured by numbers of plan failures (re-planning requests, quality of the plan, plan stability, etc.) in the course of the execution.

The objectives are described in more details and with solution directions in the dissertation plan in Section 4.

1.2 Approach

In general, the approach is based on leveraging of the well-known agent knowledge structures – the social commitments (see Section 2.2.1) – for improvement of the robustness of the plans. The plan is on a particular level of the planning process (see Section 2.1.2) transformed into the commitment form. The transformation is based on a substitution of the planning operators (or actions) by the commitments. Since the commitments are richer structures than the classical plan actions it can contain rules what has to be done to fulfill the commitment. The fulfillment of the commitment do not necessarily imply to do the hard work, the agent can delegate the commitment to another agent, or simply decommits under the applicable conditions. The particular transformation process can enrich the commitments in several ways, e.g. it can add various decommitment rules into various commitments, or can flexibly relax the commitment constraints. The rules can describe arbitrary complex commitment handling process, not only in the context of one commitment, but in the scope of the complete commitment plan and even in the scope of more than one agent. The execution model is then based on the execution of the commitments or a higher BDI[1] structures. By the execution, we understand any process of agent reasoning involving the commitments. More particularly, it can be a program in an agent programming language, a logic-based reasoner, or a rule-base description.

The approach is described in more details in Section 3 and in the dissertation plan in Section 4.

[1] Beliefs–Desires–Intentions formalism – see Section 2.2

Chapter 2

State of the Art

This chapter gives an overview of the three research fields used in the context of the proposed distributed planning by means of social commitments. The first field consists of the plan problem representation and formalization, the planning approaches, and the principles of the distribution of the planning processes. The Section 2.2 describes the state-of-the-art plan execution models based on various planning methods and agent-based reasoning (using the BDI formalism and social commitments). The well-known commitment structures and formalizations are summarizes in Section 2.2.1.

2.1 Distributed Planning

The roots of the automated planning reach the very beginnings of the artificial intelligence research and similarly, *distributed problem solving* (DPS) and *planning* (DP) are inherent parts of the scientific field of distributed AI. Edmund H. Durfee introduces his Distributed Problem Solving and Planning [9] by a statement that the DPS is a subfield of distributed AI emphasizing the collective effort of the agents to solve the common problems. In cases, the problem is to construct a plan we are talking about *distributed planning*.

The basic constructs of planning are *actions/operators*, *states*, and *goals*. The actions are elements of a plan and define how to affect the environment to reach the goals from an initial state. The states describe the environment in particular points. The goals are final states required by the planning problem. The operators are parametrized uninstantiated actions. A *planning problem* is defined by an initial state, one ore more goal states, and set of operators and actions. The result of a planning process is a *plan* which is a sequence of actions that after its execution changes the environment from the initial state to the goal state.

There are also basic phases of the planning process including *synthesis*, *inspection*, *modification*, *execution*. The first phase describes the forming of a plan. The inspection phase can provide a validation and verification of the plan, considering the requested requirements. In cases, the plan become inconsistent with the environment it has to be modified typically in form of re-planning, repairing, or merging with other plans. Finally, the plan controls a behavior of entities in the environment in the execution phase.

2.1.1 Planning Problem Representations

Each planning problem has to be appropriately represented. In [11], three main groups of representations are presented: classical, set-theoretic, and state variable representation. Each problem

in one of the representations can be converted to any other. The conversion is both time and space polynomial for all cases except the conversion to the set-theoretic description which is exponential.

The classical representation uses predicate logic. Objects in the environment are substituted by constant symbols, predicates describe relations among the objects, and a particular state is a set of ground atoms. The actions are instances (by substitution) of operators and an operator in the classical representation is a triple

$$\begin{aligned} &o = (\mathsf{name}(o), \mathsf{precond}(o), \mathsf{effects}(o)), \\ &\mathsf{name}(o) = n(x_1, \ldots, x_k), \end{aligned} \tag{2.1}$$

where the name is consisting of an operator symbol n and its parameters in form of variable symbols x_i. The preconditions $\mathsf{precond}(o)$ are set of literals which must hold in order to use the operator and the effects $\mathsf{effects}(o)$ are literals that become true after usage of the operator.

The set-theoretic representation is based on the propositional logic. Similarly to the classical representation, objects including their relations are represented by propositions. A state is a set of propositions that are true for the particular point of the environment. In contrary to the classical representation, an action is defined as a triple

$$a = (\mathsf{precs}(a), \mathsf{eff}^+(a), \mathsf{eff}^-(a)), \tag{2.2}$$

where the action is applicable in a state s only if $\mathsf{precs}(a) \subseteq s$ holds for the preconditions. The transition into a new state can be described as $s' = (s \setminus \mathsf{eff}^-(a)) \cup \mathsf{eff}^+(a)$ if precondition holds and $\mathsf{eff}^-(a) \subseteq s$ holds.

The last basic representation, similarly to the classical representation, uses ground atoms to describe properties that do not change, but it uses state variables for changing properties. A state variable of an object p is represented by an usual form $\mathsf{var}(p)$. An operator is a set of variable changes defined by action parameters.

The basic categorization of the *plan representation* is based on the linearity of the plans. A total-order plan is a ordered set of actions. The actions are executed sequentially and no actions can be processed in parallel. On the other hand, a partially-order plan is a unordered set of ordered sets of actions which enables an independent working on several subgoals (it also enables a parallel processing of more parts of the plan simultaneously). The non-linear plans need a representation for an additional constraint describing a possible collisions of the parallel sub-plans – a *causal link* [8]. The links simply describe an exclusivity of two or more actions in one state. Another categorization separates plans into two groups by a degree of how they are *instantiated*. A partially instantiated plan need another further planning process in order to be fully instantiated. Only a fully instantiated plan is directly executable.

2.1.2 Planning Techniques

The planning approaches (or more tangibly – algorithms) are usually organized into two basic groups: classical planning, and neo-classical planning. Classical planning is based on well-known AI techniques of *state-space search* (depth-first search, breadth-first search, best-first search, A* algorithm and others). According to the definition of the state-space, we can talk about planning in a *space of the operators* (it is searched a sequence of operators by traversing the operator space), or in a *space of plans* (the search goes over partially instantiated plans until a fully instantiated plan is found). Moreover, the searches based on the space of operators can be *forward-chaining* (the search begins at the initial state), *backward-chaining* (the search begins at the goal state), or *mixed* (the search goes from both sides simultaneously).

Neo-classical techniques include planning graphs, planning using hierarchical task networks (see more in Section 2.1.2.2), planning based on logical satisfiability (SAT), and planning using techniques for solving of constraint satisfaction problems (CSP). A planning graph used in GraphPlan planner is a data structure enabling effective search for a solution. The planning graph is not a classical space-state structure of operators or plans. A plan in a planning graph is a flow, in the sense of a graph flow. The planning process is based on the two iterative phases: planning graph extending and search in the planning graph. The extending phase is called until the search phase finds a solution. SAT planning is based on the fact the planning problem can be transformed into a propositional satisfiability problem. After such a transformation, we can exploit the known solvers of the SAT problem. A similar approach is used in CSP based planners. The only difference is in the transformation into the constraints-based representation.

There are many other approaches based on several principles from other research fields. An important field related to the scope of this work is planning in dynamic and/or uncertain environment. There is a wide range of approaches, for example probabilistic planning – MAXPLAN [21], contingency planning – system CIRCA [26], planning under uncertainty [4] which include solvers based on Markov Decision Processes, model checking, heuristics and meta-heuristics, modal and multi-modal logics, and others. These approaches focus on creating plan alternatives to avoid uncertainty, in contrary to the commitment based planning where the emphasis is put on individual commitments and its decommitment strategies.

The current most advanced planner is the FF-planner [12] developed by Jörg Hoffmann. It has several extensions: Conformant–FF, Contingent–FF, and Probabilistic–FF. The first extension adds an ability to handle initial state uncertainty, the second extension enables planning in partially observable environment, and the last one works with probabilistic Bayes Nets and considers probabilistic effects.

A more detailed description of the most relevant planning techniques for the scope of this work is presented in the following two subsections.

2.1.2.1 STRIPS

STRIPS (Stanford Research Institute Problem Solver) is a linear automated planner developed by Richard Fikes and Nils Nilsson [10]. The planner uses the set-theoretic representation, each operator consists of a operator symbol with a list of the parameters and three groups of expressions which are called: Preconditions (referring to $\mathsf{precs}(a)$), Deletions (referring to $\mathsf{eff}^-(a)$), and Additions (referring to $\mathsf{eff}^+(a)$). The semantics of the sets fully corresponds with the explanation of the set-theoretic representation in the previous section.

The main principle of the STRIPS planning process is an iterative composing of a successful proof in a model describing a state. The operators are analyzed and searched for an operator whose Additions-list contains only propositions which do not corrupt the partial proof or even complete the proof. Such operator is stated as a relevant and after a substitution of the parameters it is included into the (partial) proof.

The inner loop of the STRIPS process can be described in four steps:

1. Select a subgoal and check if it is satisfied in the current model. If it is, go to step 4.
2. Choose as a relevant operator whose Additions-list allows the incomplete proof of step 1 to be continued.
3. The appropriately instantiated precondition formula of the selected operator defines a new subgoal. Go to step 1.

4. If the subgoal is the main goal, finish. Otherwise, create a new model by applying the operator whose precondition is the currently established subgoal. Go to step 1.

The principle can be also explained as a boolean tree forming. The first step generates the relevant operators and second step chooses one of them. The third one generates new level of the tree. And the last one re-maps the tree to a new model and thus to a next state.

The output of the STRIPS planning process is an ordered sequence of operators (that can be substituted by the actions) and forms a plan to the goal state.

2.1.2.2 HTN

In the approaches based on the hierarchical task networks (HTN) [32], the planning process is provided with a prearranged programmer-made network of *tasks* and decomposition *methods*. In general, the network defines a full domain-specific knowledge of the environment. The tasks can be of two kinds: a *primitive task* (corresponding to an action) and *non-primitive task* (corresponding to an non-instantiated sub-plan). Each non-primitive task has a list of methods which are essentially ordered sets of other tasks which the task can be refined into. Additionally, there are usually other constraints in the network including task precedence, preconditions, parameters, variables, mutual exclusions and others.

The planning process starts with one or more initial non-primitive tasks. For each task, it is searched a method which is not in a conflict with the current plan. The method is used for decomposition of the task into more refined ones. If there is no other non-primitive task, the process has successfully found a plan. If there is a non-primitive task, the process is run recursively. If there is no applicable method for the non-primitive tasks, the planner backtracks and passes to a different decomposition.

The planning process of a HTN planner can be described in following steps:

1. The initial tasks are inserted into the plan.
2. If there are only primitive tasks, the planning process is finished and found the plan.
3. Select a non-primitive task in the current plan.
4. Select a method and refine the selected non-primitive task using the methods' subtasks.
5. Check the conflicts in the plan and run backtracking process if needed.
6. Go to step 2.

The complete decomposition contains only primitive tasks (actions) and thus it describes the final plan. It can be both totally ordered, or partially ordered (e.g. SHOP2 [27], or I-Plan [39]). The HTN planning can also be described as a plan space pruning [13].

2.1.3 Distributed Planning Techniques

The deliberative agents often need to solve collective problems that require planning. Considering such a situation, it is quite obvious we need to solve the planning problem in a distributed manner. In [9], there is an overview of techniques solving such a problems.

One of the related strategies from the field of DPS is a *task sharing* approach. The principle is based on passing of tasks from a busy agent to a vacant agent(s). The process can be summarized according to [9] in four basic steps:

1. Task decomposition: The tasks of the agents are decomposed into subtasks and subtasks which can be shared are selected.

2. Task allocation: The selected tasks are assigned to the vacant agents or agents which ask for them.

3. Task accomplishment: Each agent tries to accomplish its (sub)tasks. The tasks which need further decomposition are recursively processed and passed to other agents.

4. Result synthesis: The results of the tasks are returned to the allocating agent since it is aware of the way how to use it in the context of the higher tasks.

From the perspective of DPS, the task allocation and the result synthesis are the most crucial parts. On the other hand, from the planning perspective, the other two phases are more important. The allocation problem is usually solved by contraction and negotiation techniques which implies problems related to the resource allocation domain, e.g. cross-booking, over-booking, backtracking, and others. In the allocation phase, it also emerges a hierarchy of the agents, which may not be effective in heterogeneous multi-agent systems. In such systems, each agent has other abilities and thus a managed organizational structure is more effective [6], [30], [28]. The result synthesis can be ambiguous in sense of heterogeneous solvers in the community. For such cases, a result sharing techniques can be exploited. They improve factors like confidence (more results of one problem can be compared), completeness (one results can complete the other), precision (several parts of the results can have different qualities), timeliness (a quicker but worse solution can be in some cases superior to a better but late solution).

The problem of distributed planning can be described in one of the following forms: *centralized planning for distributed plans*, *distributed planning for a centralized plan*, and *distributed planning for distributed plans*. In the first case, the problem is to generate a set of plans for particular agents in the environment. Since the agents can work simultaneously, the plans need to be executed in parallel. This requirement is fulfilled in partially ordered plans as the ordered sets of actions act as a plan for particular agents. An example can be a HTN planner where decomposition can involve other agents [45]. The distributed planning for a centralized plan can be described as a cooperative planning of various special planners. In complex environments, it is a well-founded assumption that one planning principle is not enough. An effective heuristic can be known for one problem and an expressive representation for other. In such a case, the problem can use different planners for diferrent sub-problems. An example can be a STRIPS planner planning a high-level overall plan of and a special path planner which refines movement actions. The most challenging approach – the distributed planning for distributed plans – is a synergy of the two previous principles and it is the most suitable for the multi-agent environment. The multi-agent communities are usually heterogeneous which implies each agent is able to solve various problems. However, the agents have also a common goals and need to cooperate. Therefore, the key process that forms the plans is the fuse of the particular sub-plans of the agents. Such a process is denoted as a *plan merging*. The agents have to find points in their plans by negotiation to preserve the consistency of all the other plans.

2.2 Agent-based Plan Execution Models

Each plan, resulting from the planning process, has to be appropriately executed in the environment. An execution model describes the process how can an agent invoke the plan actions (regardless its particular form) and how the actions describe the agent's behavior. This section

BDI System	HTN System
goal-plan hierarchy	task network
goals	abstract tasks
plans	methods

Table 2.1: The mapping of BDI and HTN elements according to [31]

summarizes the state of the art in the field of planning techniques using agent-based reasoning techniques and vice versa. The Section 2.2.1 considers the agent social commitments as a preferred approach used for plan execution for the scope of the dissertation research project.

One of the first works which describes the mapping between reasoning in BDI and HTN planning is a comparative study [31] written by Lavindra de Silva and Lin Padgham. The essential observation is the principle of *decomposition* of high-level tasks (or goals) into more specific ones in both concepts and a fact the decomposition methods are defined directly by programmers. According to the authors, there had been known an *interlingua* for sharing of the informations between the two concepts, but had not existed a mapping between BDI and HTN systems. The observation is not so obvious as it could seem to be since the HTN planning is used exclusively before the execution, in comparison with BDI which controls the agent during the execution. Additionally, there is a difference in the backtracking process. As the execution in a BDI system is taking place in real time, backtracking is usually triggered by an non-deterministic effect (e.g. action failure, reading of new information, and so on) and take place also in the course of the execution. On the other hand, the backtracking process in HTN planning is caused by inconsistent plans after using of all possible methods for one abstract task. The process is triggered during the planning phase that is done before the execution. The corresponding elements of the concepts are listed in Table 2.1.

In 2005, the same authors proposed an integration of a HTN planner and a BDI system on a different basis. They described in [36] a integration mechanism – *on-demand planning* where the programmer can specify points in the BDI hierarchy which the planner should be invoked in. In the other words, if there is no prearranged applicable BDI plan, the HTN planner can be started and can generate an alternative to the statical BDI goal-plan hierarchy. The novelty of the approach is especially in the principles of the planning instantiation and initialization which uses context data of the BDI system and the current situation of the agent.

Both mentioned works were more or less practical and fundamental. The formalization of the principles comes in [34]. The work introduces an agent programming language called CanPlan which is based on the Can [47] language and HTN planning by means of on-demand planning from [36]. The corresponding entities of the systems were extended as listed in Table 2.2. The language is based on a following grammar:

$$\begin{aligned} P ::= & \mathit{nil} \mid \mathit{act} \mid ?\phi \mid +b \mid -b \mid !e \mid P_1; P_2 \mid P_1 || P_2 \mid P_1 \triangleright P_2 \mid \\ & \mathsf{Goal}(\phi_s, P, \phi_f) \mid (|\psi_1 : P_1, \ldots, \psi_n : P_n|), \end{aligned} \tag{2.3}$$

where nil is a program terminator, act is a primitive action, $?\phi$ and $!e$ are tests for a condition or an event respectively, using $\pm b$, it can be added or removed a belief, $P_1; P_2$ describes program sequencing, $P_1 || P_2$ parallelism, in $P_1 \triangleright P_2$ is a program P_2 executed only if P_1 fails its execution, the list of $\psi_1 : P_1$ describes guarded plans executed under ψ conditions, and finally in the Goal, the condition ϕ_s describes required state, P is a procedural program to fulfill ϕ_s, and ϕ_f is a condition under which the goal fails.

The Goal correlates with a commitment and under the consideration of the recursive form in

BDI System	HTN System
belief base	state
plan library	method library
event	compound task
action	primitive task
plan-body/program	network task
plan rule	method
plan rule context	method precondition
test $?l$ in plan-body	state constraints
sequence in plan-body	ordering constraint $\prec$
parallelism in plan-body	no ordering constraint

Table 2.2: The mapping of BDI and HTN elements according to [34]

expression 3.3, it could be transformed into a commitment

$$(\mathsf{Commit}\ A\ true\ \phi_s\ \{(\mathsf{Commit}\ A\ false\ \phi_f\ \emptyset)\}), \tag{2.4}$$

In [35], the authors have extend the formalism by more expressive description of the goals which is based on the basics of the commitments. The motivation was to describe all well-known forms of the commitments: *blind minded*, *single minded* and *open-minded*. They have introduced a *declarative active goal* which is inductively defined for program P and declarative goals in form $\mathsf{G}(\phi_s, \phi_f)$ as follows:

$$\mathcal{G}(P) = \begin{cases} \mathcal{G}(P_1) & if P = P_1; P_2 | P_1 \triangleright P_2 | \mathsf{Plan}(P_1), \\ \mathcal{G}(P_1) \cup \mathcal{G}(P_2) & if P = P_1; P_2 || P_1, \\ \mathcal{G}(P_1) \cup \{\mathsf{G}(\phi_s, \phi_f)\} & if P = \mathsf{Goal}(\phi_s, P_1 \triangleright P_2, \phi_f), \\ \emptyset & otherwise, \end{cases} \tag{2.5}$$

The last contribution by Silva and Padgham in [19] evolves the integration of the BDI and HTN systems even more by so called *hybrid-solutions*. According to the authors, an ideal hybrid-solution is able to plan with abstract operators, i.e. BDI event-goals. However, the problem is the plan needs primitive actions to be executable by the agents. The work presents an approach how to obtain an abstract plan and how to fix several points of the plan to be instantiable by primitive actions.

A framework for fuse of the HTN planning and mental attitudes of a BDI agent is presented in [46]. The framework reconciles I-X [39] platform (which includes I-Plan, a HTN planner) and its underlying <I-N-C-A> [40] representation. The agents adopt an *activity-centric* view which is implied by <I-N-C-A>-based reasoning of the agents. The reasoning is done on a knowledge structure including goals, action consequences, constraints and other knowledge structures. Additionally, the agent's state and behavior is described in a form of a plan. The extension of <I-N-C-A> by mental attitudes provide a possibility to represent intentions as activities at various levels of abstraction. The semantics is based on the HTN procedures. The procedures describe how the specification of the agent, together with the perception, result in executable actions.

An <I-N-C-A> object based on the <I-N-C-A> model is a 4-tuple

$$(I, N, C, A), \tag{2.6}$$

where I is a set of *issues*, N is a set of *activity nodes*, C is set of *constraints*, and A is a set of *annotations*. Issues are elements of the planning process that has to be considered. Thus only

BDI System	<I-N-C-A> System
beliefs	constraints
desires	*implicit* in reasoning elements
plans	methods

Table 2.3: The mapping of BDI and <I-N-C-A> elements by [46]

a empty set of issues is consistent with a formed plan (there are no other issues to consider). Formally, an issue is a set of activities

$$l : M(O_1, \ldots, O_n), \tag{2.7}$$

where l is a label of the issue, M is a modificator of the plan, and O_i is an i-th argument of the modificator (it can be in any form of I, N, C, or A)[1]. Nodes are activities[2] to be performed in the plan. The definition contains an activity α, and an extra duration of the action (beginning at τ_b and ending at τ_e):

$$l : \alpha(o_1, \ldots, o_n)@[\tau_b, \tau_e]. \tag{2.8}$$

Constraints[3] must be satisfied in the plan and one is defined as follows:

$$l : c(v_1, \ldots, v_n), \tag{2.9}$$

where c is the relation symbol and v_i is a relation variable. The mappings of the BDI concepts and <I-N-C-A> models are summarized in Table 2.3.

An approach based on another than HTN planning paradigm is introduced in [24]. It uses a GraphPlan planner [3] and an extension of BDI called the X-BDI agent model [25]. Since the classical BDI models are not directly executable (there is no simple implementation of the underlying multi-modal logics), X-BDI is designed to be directly transformable into a programming language. X-BDI uses a formalism which has its own implementation. The implementation is called *Extended Logic Programming with explicit negation* (ELP) and uses the *Well-Founded Semantics extended for the explicit Negation* (WFSX). As a procedure for derivation in the logic is used *Selected Linear Derivation for extended programs* (SLX) [1]. According to the authors of the framework, X-BDI uses ELP's ability to deal with contradiction to implement a variety of *non-monotonic reasoning*.

All previous frameworks and formalizations do not explicitly consider a dynamic and incompletely known environment, although such a environment is very relevant for planning domains mentioned in Section 3.2. A formal model based on such a premises has been developed in terms of research done in [20]. The model assumes the agents' tasks and behaviors are expressed in a form of high-level non-deterministic concurrent programs. As the work shows, the result of the planning process must take into account the non-determinisms, however the plan must be deterministic (so as it can be executed). This contradiction is solved by deterministic conditional plans which can be successfully executed against all a priori known non-deterministic effects in the environment. Such effects also include sensing as it can be understood as a unpredictable from the agent's perspective. The conditions in the plan act as points of real-time decision for the agent. Such a planning model is classically called *contingent* (or *contingency*) planning.

[1] An example of an issue/meta-activity `achieve` for abstract achieving of a world-state p in time τ is l : `achieve`(p, τ).

[2] An example of an activity can be l : `build-house`(A, l, t) where agent A has to build a house of type t at location l.

[3] An example of a constraint can be l : `before`(a, b) that says the proposition a must happen before b.

The novelty of the model introduced in this work is in the formalization in a form of an agent programming language (APL) with a *transition semantics*. The programs for the agents, the environment, and the planning mechanisms are expressed in the ConGolog language [7]. Programming constructs of the language are following:

α	primitive action
$\phi?$	wait for a condition
$\delta_1; \delta_2$	sequence
$\delta_1 \vert \delta_2$	non-deterministic branch
$\pi\ \vec{x}.\delta$	non-deterministic choice of argument
δ^*	non-deterministic iteration
if ϕ **then** δ_1 **else** δ_2 **endIf**	conditional
while ϕ **do** δ **endWhile**	while loop
$\delta_1 \parallel \delta_2$	concurrency with equal priority
$\delta_1 \rangle\rangle \delta_2$	concurrency with δ_1 at a higher priority
$\delta^{\parallel}$	concurrent iteration
$\langle \vec{x} : \phi \rightarrow \delta \rangle$	interrupt
$\vec{p}(\theta)$	procedure call

In the used formalization, δ denotes a program, ϕ is a conditional proposition, $\vec{x}$ is a vector of variables, π is a non-deterministic binding of variables and a program, and θ is a vector of procedures. The planning mechanism using the ConGolog language is based on sequencing of actions by two procedures which are called recursively. The procedures perform an iterative deepening search for a conditional plan with a defined final goal.

2.2.1 Social Commitments

The plan execution model introduced in Section 1.2 is based on an agent knowledge structure denoted as a social commitment. This section descibes the state-of-the-art formalizations and definitions of the social commitments.

In the multi-agent world, one of the key characteristic of the agents is their mutual cooperation. With the help of the cooperation, the agent community can lead to common goals and profit. The principle of the cooperation is based on compliance on agents' particular obligations, which are usually established via multi-agent negotiation. In the terms of agents, such a obligation is a knowledge-base structure describing an agent's (or agents') *commitment* to achieve a specific goal. In other words, if the agent adopts an intention to fulfill a goal, it commits to the goal [48]. In some cases, a commitment can be requested by other agent(s), which means it describes a bond between one or more agents (which are committed) on one side and agent or agents (which are requesters) on the other side. Such a bond is called – *social commitment* [49], [37].

Michael Wooldridge in [48] defines the commitments formally as follows:

$$\begin{aligned} &(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda), \\ &\quad \lambda = \{(\rho_1, \gamma_1), (\rho_2, \gamma_2), \ldots, (\rho_k, \gamma_k)\}, \end{aligned} \tag{2.10}$$

where A denotes a committing agent, ψ is an activation condition, φ is a commitment goal, and λ is a convention. The convention is a set of tuples (ρ, γ) where ρ is a decommitment condition and γ is an inevitable outcome. The convention describes all possible ways how the commitment can be dropped. Generally speaking, the actor A has to transform the world-state in such a way that the φ goal becomes true if ψ holds and any γ has not been made true yet. The actor is allowed to drop the commitment if and only if $\exists i : \rho_i$ which is valid. A decommitment

is allowed provided that γ_i is made true. A formal definition in modal logic (working with the models of mental attitudes like Believes, Desires, Intentions, [38], and temporal logic where the operator AG denotes an the inevitability and operator $\curvearrowleft$ denotes the temporal until) follows as defined in [48]:

$$\begin{aligned}
&(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda) \equiv \\
&\quad((\mathsf{Bel}\ A\ \psi) \Rightarrow \mathsf{AG}((\mathsf{Int}\ A\ \varphi) \\
&\qquad\wedge(((\mathsf{Bel}\ A\ \rho_1) \Rightarrow \mathsf{AG}((\mathsf{Int}\ A\ \gamma_1))) \curvearrowleft \gamma_1) \\
&\qquad\ldots \\
&\qquad\wedge(((\mathsf{Bel}\ A\ \rho_k) \Rightarrow \mathsf{AG}((\mathsf{Int}\ A\ \gamma_k))) \curvearrowleft \gamma_k) \\
&\quad) \curvearrowleft \bigvee_i \gamma_i).
\end{aligned} \tag{2.11}$$

This definition is used in a declarative way. Provided that whatever the agent does during a specific behavior run complies with the above defined commitment, the expression 2.11 is valid throughout the whole duration of the run. In the outer part

$$(\mathsf{Bel}\ A\ \psi) \Rightarrow \mathsf{AG}(\ldots) \curvearrowleft \bigvee_i \gamma_i, \tag{2.12}$$

the implication enforces the agent to adopt the inner (intentional) part only if it believes ψ until any of γ_i outcomes is fulfilled. It means the commitment can be dropped even without any participation of the agent A so far as the γ_i was fulfilled by another agent.

The intentional part of the commitment invokes the main intention to comply φ and additionally it can turn on the convention rule (ρ_i, γ_i) (by similar principle as the outer part).

Another commitment definition and formalization can be found in [37] whose author is Munindar P. Singh. Specifically, it concerns social commitments and the definition is following:

$$c = \mathsf{C}(x, y, G, p). \tag{2.13}$$

The expression denotes a commitment from a debtor agent x toward a creditor agent y. G describes a context group of the commitment and the proposition p describes a discharge condition.

Here, if the debtor agent accepts the commitment, it is obliged to comply the discharge condition. If it become true, the condition also fulfills the commitment as a whole. The creditor needs not to be the direct beneficiary of the commitment, but as the author analyses, it usually is. The context group describes the conditions under which the commitment was established. More precisely, the context described the norms or conventions. Their usage is required in the case of conflict between the debtor and creditor agents. Such a group usually consists of other agents which mediates mentioned conditions and acts as a arbiters of the conflict (the author points the analogy with courts and delegation of the authority on a small group of specialized actors).

In [37], there are also defined operations on the commitments:

- *Create* instantiates a commitment,
- *Discharge* satisfies the commitment,
- *Cancel* revokes the commitment,
- *Release* essentially eliminates the commitment,
- *Delegate* shifts the role of debtor to another agent,
- *Assign* transfers a commitment to another creditor.

The operations are invoked by the agents and they influence the commitments in appropriate ways. In [50], it is the proposed commitment form used for definition of an abstract formal interaction model among the agents. The model is based on commitment patterns described in temporal logic. Similarly in [51], the commitments and its operations are used for formalization of agent protocols. The formalization is based on event calculus and the adoption of the commitments enables flexible execution paths of the protocols. Modeling of the exceptions in the commitment protocols is shown in [23]. Avali and Huhns show in [2] approach to multi-agent decision making, based on the commitments defined in [37]. The proposed commitments are based on BDI_{CTL^*} framework which is a bridge between BDI and computational tree logic. Other view on the problem of branching time in commitments is formalized in [22] that is a work of the same author. Here, the temporal commitments are introduced and time intervals are used as the apparatus for their description.

Considerably different view and formalization is proposed in [5] where the commitments are based on the deontic logic. The authors show the principle on an example of two actions *propose* and *accept* that form the commitment. The actions are adopted by two agents, where the committed agent uses the action accept, and the requesting agent uses the action propose. Formally, the actions are of the form:

$$propose_j(\alpha for \beta), \tag{2.14}$$

where j is the requested agent and α denotes requested action in return for β. Acceptance of the proposal has a similar form:

$$accept_i(\alpha for \beta), \tag{2.15}$$

where i is the accepting agent. In the formalism introduced in the work, the commitment can be described subsequently:

$$[propose_j(\alpha for \beta)]_i[accept_i(\alpha for \beta)]_j(O_{ij}(\alpha) \wedge O_{ji}(\beta)). \tag{2.16}$$

According to [5], "the formula expresses that the obligation of the proposing agent is conditional on the acceptance of it by second agent, and that the obligation of the accepting agent is a direct consequence of the acceptance."

Beside the theory of commitments, there is a wide area of research dedicated to contracts and relative themes. The principles are based on the leveled commitments introduced by Sandholm and Lesser in [33]. Contracts are usually richer and based on several ontologies. In [29], it is shown an ontology for an agent contracting language which uses commitments on level of behavioral control mechanisms. In contrast to commitments, contracts can wrap more concepts including agent roles, agent parties, contexts of the contract, penalties, prices, and others.

Chapter 3

Current Contributions

The recent student's work evolves the state of the art research especially by three concepts. The first student's contribution is an extension of the commitment definition in [15]. The extension is based on the original Wooldrige's [48] definition of a commitment. A design of an architecture corresponding to the planning of commitments approach was the second student's contribution to the discussed research field [15]. Works analyzing the impacts of particular decommitment rules are the last contribution. The decommitment rules define the plan execution model and that affects the successfulness of the fulfillment of the agent goals [42], [41].

3.1 Extended Definitions of a Commitment

The extension is based on the definition in the expression 2.11. The original commitment form (with bindings to another commitment) is depicted in Figure 3.1.

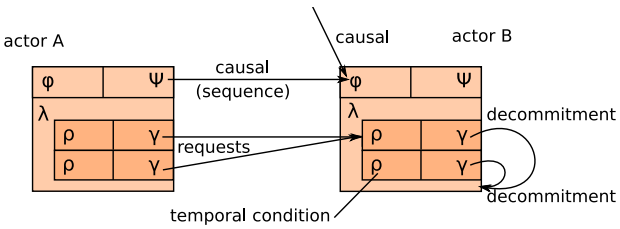

Figure 3.1: Commitments and bindings - the actor A's commitment influences the actor B's commitment using the causal (sequential) link, the link is described using the ψ and φ clauses. The actor B's commitment is influenced by external causality too. The actor B's commitment can be decommitted in two cases: either the *temporal condition* ρ becomes true or one of the actor A's rules *requests* the decommitting. The decommitment request is triggered by one of the actor A's ρ conditions.

The original definition makes a clear distinction between the commitment subject φ and the goals γ in the commitment convention set, but after analysis of expression 2.11 one can show that the goals φ and γ can be unified. The unification allows us to describe more complex decommitting mechanism and allows the agent to try out several different decommitment alternatives, based on the current properties of the environment.

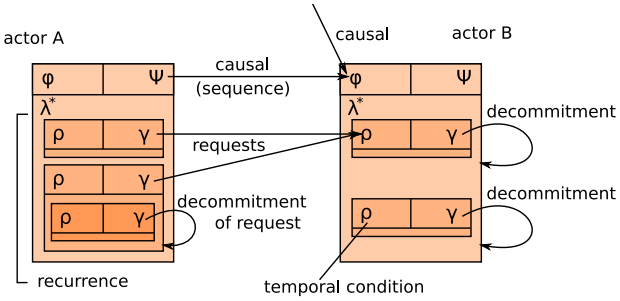

Figure 3.2: Commitment and its λ^* commitments – the Fig. 3.1. is extended by one *decommitment of request* which can be decommitted if the most inner ρ condition becomes true. De-committing of the request causes the actor B's commitment cannot be decommited by the actor A's convention goal any more. Here the recursive form enables the nesting of the inner commitment.

That is why we have introduce the recursive form of a commitment, which enables the nesting of the commitments – Fig. 3.2:

$$\begin{array}{l}(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda^*), \lambda^* = \\ \quad \{(\mathsf{Commit}\ x_1\ \rho_1\ \gamma_1\ \lambda_1^*), \\ \quad (\mathsf{Commit}\ x_2\ \rho_2\ \gamma_2\ \lambda_2^*), \ldots, \\ \quad (\mathsf{Commit}\ x_k\ \rho_k\ \gamma_k\ \lambda_k^*)\}.\end{array} \tag{3.1}$$

The formula 3.1 extends the definition in 2.11 not only by inclusion of a set of decommitment rules in each of the individual decommitment rules. It also allows the newly adopted commitments to be assigned to different actors. The delegation kind of decommitment between two agents A and B would have the following form:

$$(\mathsf{Commit}\ A\ \psi\ \varphi\ \{(\mathsf{Commit}\ B\ \rho\ \varphi\ \emptyset)\}), \tag{3.2}$$

representing that agent A can drop the commitment towards φ provided that ρ is valid and provided that B accepts a commitment towards φ on A's behalf.

This form is very expressive in the sense of the description of exceptional states. It allows us to have a branched chain of individual nested commitments for each individual situation. The recursive nature allows us to describe an arbitrarily complex protocol using only one knowledge base structure - a recursive form of the commitment. The recursive form of the commitment is thus defined as:

$$\begin{array}{l}(\mathsf{Commit}\ A\ \psi\ \varphi\ \lambda^*) \equiv \\ \quad ((\mathsf{Bel}\ A\ \psi) \Rightarrow \mathrm{A}((\mathsf{Int}\ A\ \varphi) \wedge \bigwedge_j \lambda_j^*) \curvearrowleft \bigvee_i \gamma_i).\end{array} \tag{3.3}$$

The expression 3.3 can be simply derived from formulas 2.10, 2.11, and 3.1. The basic principle is based on a fact the inner part of the expression 2.11

$$((\mathsf{Bel}\ A\ \rho_1) \Rightarrow \mathsf{AG}((\mathsf{Int}\ A\ \gamma_1))) \curvearrowleft \gamma_1 \tag{3.4}$$

is, in the fact, the same as the outer part

$$(\mathsf{Bel}\ A\ \psi) \Rightarrow \mathsf{AG}(\ldots) \curvearrowleft \bigvee_i \gamma_i. \tag{3.5}$$

Furthermore, we have used the definition to propose a basic commitment's convention set in [41], and [16]. Each decommitment rule set (corresponding to Wooldridge's commitment convention) must contain two basic rules which ensure the rationality of the agent's decision making process. These rules are based on the definition of the *open-minded commitment* defined in [48]:

$$\begin{aligned}&(\mathsf{Commit}\ A\ \varphi) \equiv \\ &\quad \mathsf{AG}((\mathsf{Int}\ A\ \varphi) \curvearrowleft ((\mathsf{Bel}\ A\ \varphi) \vee \neg(\mathsf{Bel}\ A\ \mathsf{EF}\varphi))),\end{aligned} \tag{3.6}$$

where the operator EF denotes future possibility. Thus in each and every commitment the initial decommitment in the λ^* set should be the success triggering commitment

$$(\mathsf{Commit}\ A\ false\ (\mathsf{Bel}\ A\ \varphi)\ \emptyset) \tag{3.7}$$

and the λ^* set should be always closed with a fail-safe trigger turning a violated commitment eventually off

$$(\mathsf{Commit}\ A\ false\ \neg(\mathsf{Bel}\ A\ \mathsf{EF}\varphi)\ \emptyset), \tag{3.8}$$

which is used provided that no other decommitment rule can be used and the parent commitment became unrealizable.

The triggering decommitment condition ρ in rules (3.7) and (3.8) cannot become true (as $\rho = false$), which means the agent will never intend the decommitment outcome γ according to the definition (3.3)

$$\begin{array}{lcl}((\mathsf{Bel}\ A\ \rho) & \Rightarrow & \mathsf{AG}((\mathsf{Int}\ A\ \gamma))) \curvearrowleft \gamma \\ (false & \Rightarrow & \mathsf{AG}((\mathsf{Int}\ A\ \gamma))) \curvearrowleft \gamma,\end{array} \tag{3.9}$$

nevertheless, the rule can drop the commitment using the drop-out part

$$\bigvee_i \gamma_i. \tag{3.10}$$

Moreover, this principle can be used for any drop-out rule with no need for explicit intention.

3.2 Architecture for Planning with Commitments

The architecture design was motivated by need for planning tool which would be able to reasonably work in *distributed*, *dynamic*, and *highly non-deterministic* environment. The plans suitable for the presented properties of the environment require robustness (e.g. in form of plan contingencies), plan stability (especially for comfort of the human operator), preservation of private knowledge of the agents, and finally, scalability towards numbers of cooperating agents.

The problem of automated planning in such a world can be described by several more or less separable problems which need to be solved in order to be able to plan in such a environment. The overview of the problems from [16] follows:

- **Distributed planning** – the question how to create the plans.
- **Distributed resource allocation** – the question how to instantiate the plans with particular resources (this issue is not discussed in this work).
- **Distributed plan execution and synchronization** – the question how to execute the plans in distributed manner and with respect to all proposed requirements (robustness, scalability, and others).

Planning and control of activities of individual units and actors in the scenario has been loosely structured into three levels of detail. We have recognized several layers of coordination and control:

- **Strategic layer** – the layer provides high-level planning.
- **Tactical layer** – the layer provides cooperation methods and can create new goals or adapt the goals received from the strategic level.
- **Individual layer** – the layer provides a behavior of the units towards the simulated world

The suggested coordination has been designed as a hierarchical with respect to the type of unit, area of operation and visibility. Three-layer architecture enables to separate middle- and long-term strategic planners from the real-time planning and control on the tactical and individual level.

Each layer produces particular commitments and these commitments define the plan. The commitments describe which actions should be executed (simply by defining that $\varphi = a$, where a is an action) and decommitment rules (including delegation policies and relaxation intervals).

The strategic layer uses the HTN I-X planner [39]. The planner uses an abstract sub-domain derived from the scenario domain and produces an abstract plan. This plan is instantiated using negotiation about the resources (Figure 3.3).

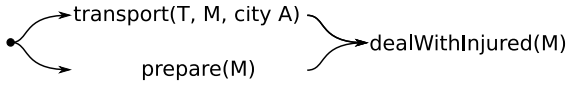

Figure 3.3: Example of instantiated strategic plan - the medic unit M was requested by the commander agent to fulfill a task: deal with the injured in city A, and it negotiated the transport with the transport unit T.

The instantiated plan is converted into commitments (Figure 3.4). The conversion process creates a commitment according to the particular plan action ($\varphi = a$) and according to forward causality links [8] of the plan.

The commitments of the tactical layer are based on strategic commitments. The layer uses negotiation to form the most suitable mutual commitments. The tactical commitments also define decommitting rules to the strategic layer and they can additionally refine strategic commitments too. They are much more refined than the strategic commitments in the sense of spatio-temporal constraints, and particular world states. The tactical commitments are most enriched by the λ^* commitments. Thus, the most important part of the decommitting / re-planning process is done by this layer (the principle of the re-planning process will be mentioned later on in violation solving algorithm).

And finally, the individual layer plans commitments for later execution. These commitments copy the tactical commitments, but some of these can be omitted (e.g. $atPosition$ in the Figure 3.4). Each individual commitment contains a decommitment request only to its parent commitment (from the tactical layer). The individual commitments have the same debtor and creditors, which means the agent commits to itself.

3.2.1 Commitment Graph

With the help of the proposed recursive commitment form (see Section 3.1), we have been able to transform the set of mutually interconnected commitments into a graph notation. The formalization is convenient especially due to a broad field of well-known principles and algorithms (e.g.

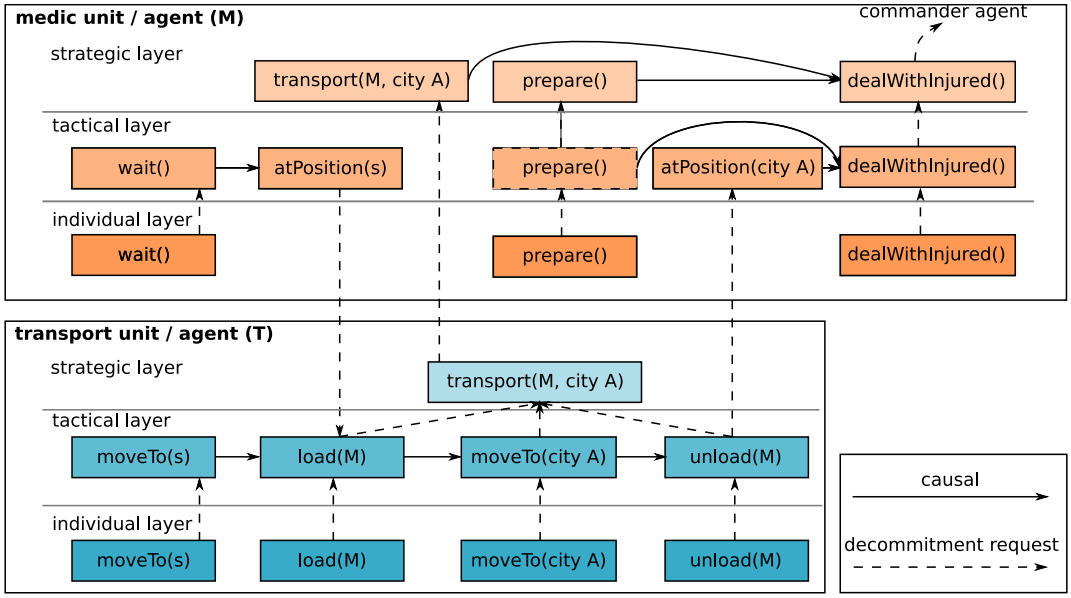

Figure 3.4: Example of commitment bindings of multi-layer architecture for two units – the medic unit M is committed to fulfill a task: deal with the injured in city A, and the transport unit T is committed to transport the medic unit M to city A. The figure shows the directions of the potential decommitment propagation among the layers of actors.

dead-lock detecting, known complexity, extensive transformations and others). Additionally, we have proposed a violation solving algorithm, based on graph representation of the commitments.

Vertices of the *commitment graph* (Figure 3.5) represent particular commitments and edges describe inter-commitment bindings (causality and decommitment requests). Formally, the commitment graph is defined as follows:

$$\begin{aligned} &\vec{G} = (V, E), \\ &M : V(\vec{G}) \rightarrow \mathcal{P}((\mathsf{Commit})), \\ &W : E(\vec{G}) \rightarrow \mathcal{P}((\mathsf{Commit})) \times \mathcal{P}((\mathsf{Commit})), \end{aligned} \tag{3.11}$$

where $\vec{G}$ is an oriented graph with vertices V and edges E. Additionally, the functions M and W map the commitments and their bindings to the vertices and edges. The set $\mathcal{P}((\mathsf{Commit}))$ stands for all possible commitments.

A graph notation can be used to describe the process of successive solving of exceptional states. The process is based on traversing through the commitment graph. The traversing starts with the first violated commitment. One of the decommitment rules is triggered (depending on the violation type). As the decommitment rule is a commitment it invokes an intention of the agent to terminate the commitment. In the case that the intention is a decommitment request, the process passes on the requested commitment (decommitment rule respectively) and starts one of the decommitment rules on the side of the requested commitment. Provided that the decommitment rule terminates the commitment without a need to request other decommitments, the process ends here and the violation is fixed.

The algorithm written in BDI pseudocode follows:

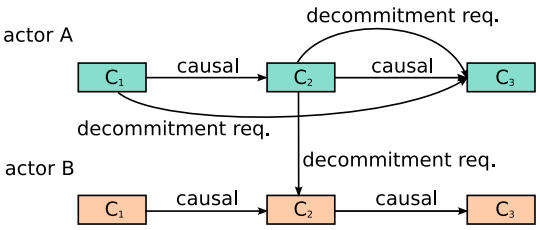

Figure 3.5: Example of the commitment graph – the *causal* links define the sequentiality of the commitments of each actor. The commitment C_3 of the actor A can be decommitted by both C_1 and C_2 commitments. The C_2 commitment of the actor B can be decommitted by actor A using the decommitment request.

`Input:` Vertex v of a commitment graph $\vec{G}$
representing violated commitment C.
`Output:` Updated graph with solved violation.
function solve(v, C, $\vec{G}$) **begin**
D := find appropriate decommitment rule in C
run D **begin**
(Int A $\gamma(D)$) $\Rightarrow$
if $\gamma(D)$ is request **then**
solve($M^{-1}(\gamma(D)(M(v)))$, $\gamma(D)(C)$, $\vec{G}$)
else
fix a subgraph induced by vertex v
from graph up $\vec{G}$
end
end
end

In the algorithm, $\gamma(D)$ denotes a goal γ of a nested commitment D (thus for this case $\varphi = \gamma$). Considering the recursive form of the commitments (expression 3.1), we can understand decommitment rule as a commitment in line

---

D := find appropriate decommitment rule in C.

Functions M and M^{-1} respectively are graph mapping functions from graph vertices to commitments and vice versa.

3.2.2 Decommitment Rules

The decommitment rules are according to the SOTA in Section 2.2.1 the ways how the commitment can be dropped. In [41], we have investigates how ordered particular rules influence the robustnes of the plans.

For the course of the plan execution, we have introduced three different types of decommitment rules based on the definition in [41]:

- *Full decommitment* (Fd) - the basic decommitment strategy is dropping the commitment. Under defined circumstances the agent is completely released from the commitment. The full decommitment decommits the original commitment if and only if the commitment goal φ is unrealizable.

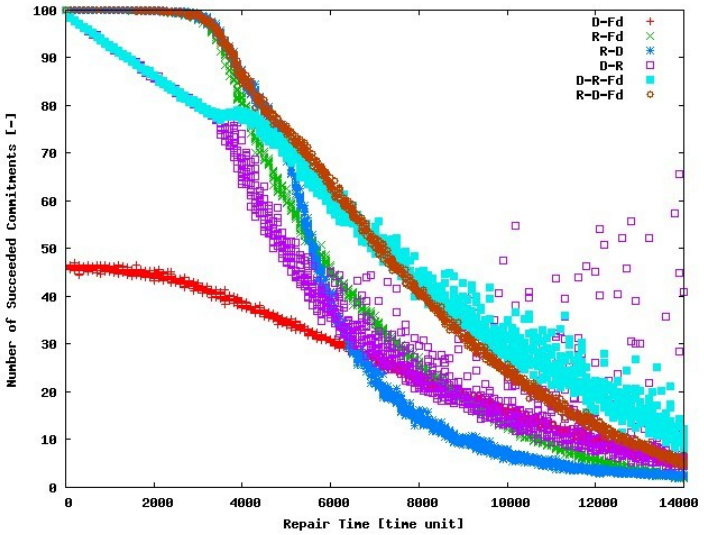

Figure 3.6: Number of succeeded commitments (and thus succeeded plans) for varying duration of the non-deterministic effect (repair time of the entitiy) for different decommitment rules settings, more details about the envritonment and the particular scenario can be found in [41].

- *Delegation* (D) - by using this type of commitments the agent shall be able to find another agent who will be able to complete its commitment on the original agent's behalf. It is possible that such a commitment will contain unbound variables representing the need to search for an agent suitable for delegation. Delegation decommits the original commitment if and only if the commitment goal φ is unrealizable and the new commitment on the other agent's side is formed.
- *Relaxation* (R) - is a special decommitment, where the original commitment is replaced with a new commitment with a relaxed condition and/or goal. The new commitment must be consistent with all other bound commitments. Provided that the bound commitment is of another agent, the relaxation must be negotiated. The agent being addressed tries to fit the requested relaxed commitment into its knowledge base and eventually use some other decommitment rules of other commitments to change it and fulfill the request. Relaxation decommits the original commitment if and only if the commitment goal φ is unrealizable, the negotiated relaxation conditions hold and the relaxed commitment is formed.

During the execution of the plan the commitments are processed. The commitment can evolve according to the plan or due to unexpected environment interactions. The ordering and presence of particular decommitment rules between the two basic decommitments (success and violation) has a non-trivial impact on the robustness of the execution (especially in an overloaded system) as shown in [41]. As shown in Figure 3.6, the best results for heavily stressed systems can be reached by the ordered set of (D, R, Fd) decommitment rules. On the other hand, the best set for more vacant systems seems to be (R, D, Fd).

The experimental evaluation proved significant improvement of plan execution performance and stability when using particular combinations of decommitment rules. The success rate of commitment execution and available resources utilization significantly increases with the size of the decommitment rule set. A detailed analysis of the relaxation decommitment rule and its utilization for the presented scenario can be found in [42].

3.3 On Contributions

In this section, a summary of all student's papers and articles related to the dissertation research project is presented. The first contribution considering the multi-layer planning architecture (Section 3.2) is presented in [15] together with a definition and formalization of the recursive form of the commitments (Section 3.1). Further formalization of the decommitment rules (Section 3.2.2) is presented in [41] together with experimental validation of their impact on the robustness of the plan execution. The impact of the relaxation decommitment rule is in detail analyzed in [42]. The experimental system I-Globe is published from various points of view in three papers. The first one is a demo presentation [17] showing the complete system on the complex disaster relief scenarios, the second one shows the system from the perspectine of team-oriented operations in [18], and the last one presented in [14] describes especially the distributed planning part of the system. Additionally, there is a publication [44] adopting the principles of planning based on the decommitment rules in the context of the <I-N-C-A> model. The last publication is an invited journal article [16] describing usage of the commitments-based planning in the multi-layer planning architecture.

Chapter 4

Dissertation Plan

The research plan is based on the four main objectives proposed in Section 1.1. Firstly, there is a summary of the work already performed according to Section 3. Secondly, the future work is described in the context of the main objectives.

In general, the accomplished work corresponds partially to all the objectives excluding the planning problem representation. The multi-layer planning architecture (Section 3.2) acts as a planning system and correlates with the *Planner* objective. It is based on the HTN planner that is used on various planning levels and by each agent per se. The planning problem representation is an integral part of the used planner. The planner has been extended to generate the plans in the form of the social commitments. The commitments form the distributed plan among the agents and their planning levels. The execution model uses the distributed commitment plan. Agents process the plan by commiting to the goals of the actions and by performing such commitments. Additionally, they can operatively change the plan through the decommitment rules. The approach has been validated in a multi-agent environment in a simulated scenario of the real-world-based disaster relief and logistic operations [17].

The future work will be based on the work already done. The multi-layer planning architecture will be extended using the concepts of the planning by means of the commitments improving the robustness of the plans. The planning problem needs an unifying formalization which will be expressive enough to describe the planning problem, the planning process, and the commitment structures describing the plan. The execution model will be extended to be able to invoke the planning process by itself and thus improve the robustness even more as the planning process could respond to the particular situation in the environment. Finally, the approaches will be validated on an extension of the current experimental scenarios. The summary of the objectives with more detailed description follows:

1. **Planning problem representation** – The representation of the planning problem should be described in the same formalization as the commitments for the possible direct planning with the commitments. There are similar formalizations summarized in Section 2.2 which are primarily designed for usage in pure BDI, however another formalization could be more suitable for the description of structures based on the commitments. The other part of the objective is to find (and eventually extend) an formal framework based on the mentioned formalization. Similarly to the X-BDI (in Section 2.2), a formal framework enabling its future implementation will be needed for validation and experimental purposes. The framework could be based on one of those introduced in [43], or in [35].

2. **Planner** – The planner should be based on the contingency planning approaches and

methods introduced in [34], but it will extend the field to solve problems of the complex, dynamic, and non-deterministic environment. The plans could contain prearranged reactive contingency branches, described in the form of the commitments. The contingencies could be utilized among all agents since the commitments can be delegated to other agents. The re-planning process will be invoked in a case of an unsolvable situation (eventuality which was not described by the contingencies). Furthermore, the planner should adopt a self-containing process of planning supported by the plan form of the social commitments. The planning process could contain not only the tasks which the agent has to fulfill, but also tasks invoking recursively a new planning processes which can completely rebuild the commitment base and thus the future behavior of the agent.

3. **Plan execution model** – The principle of the plans based on the commitments is similar to the concept of the on-demand planning proposed in [36], but the essential difference is in the structure holding the plan. In the on demand planning the plan is only a complex decomposition of a BDI task, but in the commitment based planning, the complete plan could be defined by the commitments. It implies the execution process of the agent would be fully controlled by the planning process whereas the planning process is invoked (and controlled) by the execution process. Such plans should be robust against the environment, since the complete execution and planning process could be changed according to the actual conditions.

4. **Experimental validation in a multi-agent prototype** – The approaches will be experimentally evaluated in the multi-agent environment similar to the already used one. The experiments will be appropriately adjusted and extended to be able to plausibly validate the new concepts. Since the robustness is the main criterion, the experiments will be designed to measure successfulness of the plans for various parameters of the environment. The successfulness of the plans can be simply measured by numbers of failures (re-planning requests, quality of the plan, plan stability, etc.) in the course of the execution. The experiments will include both synthetic and real-world-inspired scenarios. The synthetic scenarios can for example directly measure the effect of particular decommitment rules and their mutual interactions. On the other hand, the real-world scenarios show the overall effectiveness of the approach. It should reveal non-trivial regularities caused by the complexity of the distributed planning process and commitment structures with the decommitment rules.

The fundamental contribution of the research will be deepening of the state-of-the-art findings in the area of the multi-agent planning and a design of a planning technique which enables robust planning in the dynamic, non-deterministic and distributed environments.

Chapter 5

Conclusion

The work summarizes the research field of planning (with its distributed form) and gives the state of the art of the concept of social commitments and agent-based plan execution models based on the commitments and BDI-based agents. The overview is also supplemented by the student's contributions. The first contribution is an extension of a well-known formalization of a commitment and the second one is a design of the layered planning architecture, involving HTN planning and commitments-based plan representation, and the last one describes the basics of the decommitment rules and commitment plan execution. The work is concluded by the dissertation plan.

Bibliography

[1] J. J. Alferes and L. M. Pereira. Reasoning with logic programming, 1996.

[2] V. R. Avali and M. N. Huhns. Commitment-based multiagent decision making. In *CIA '08: Proceedings of the 12th international workshop on Cooperative Information Agents XII*, pages 249–263, Berlin, Heidelberg, 2008. Springer-Verlag.

[3] A. L. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:1636–1642, 1995.

[4] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[5] J. Broersen, M. Dastani, Z. Huang, and L. van der Torre. Trust and commitment in dynamic logic, 2002.

[6] D. D. Corkill. *A Framework for Organizational Self-Design in Distributed Problem Solving Networks*. PhD thesis, University of Massachusetts, 1982.

[7] G. de Giacomo, Y. Lespérance, and H. J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.*, 121(1-2):109–169, 2000.

[8] B. Drabble and A. Tate. O-plan: A situated planning agent, 1995.

[9] E. H. Durfee. Distributed problem solving and planning, 1999.

[10] R. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artif. Intell.*, 3(1-3):251–288, 1972.

[11] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers, May 2004. ISBN 1-55860-856-7.

[12] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[13] S. Kambhampati, A. Mali, and B. Srivastava. Hybrid planning for partially hierarchical domains. In *IN PROC. 15TH NAT. CONF. AI*, pages 882–888. AAAI, 1998.

[14] A. Komenda. Distributed planning and coordination in multi-agent environment. In *CTU Workshop: 18th Annual CTU University-wide Seminar*, volume 13, CTU in Prague, Feb 2009.

[15] A. Komenda, M. Pěchouček, J. Bíba, and J. Vokřínek. Planning and re-planning in multi-actors scenarios by means of social commitments. In *Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT/ABC 2008)*, volume 3, pages 39–45. IEEE, october 2008.

[16] A. Komenda, J. Vokřínek, and M. Pěchouček. Planning and re-planning in multi-actors scenarios by means of social commitments. *Special Issue of the Web Intelligence and Agent Systems An International Journal*, 2010 (submitted).

[17] A. Komenda, J. Vokřínek, M. Pěchouček, G. Wickler, J. Dalton, and A. Tate. Distributed planning and coordination in non-deterministic environments (demo). In *AAMAS '09: Proceedings of the eight international joint conference on Autonomous agents and multiagent systems*, 2009.

[18] A. Komenda, J. Vokřínek, M. Pěchouček, G. Wickler, J. Dalton, and A. Tate. I-globe: Distributed planning and coordination of mixed-initiative activities. In *KSCO '09: Knowledge Systems for Coalition Operations 2009*, Chilworth Manor, Southampton, UK, Mar-Apr 2009.

[19] Lavindra, S. Sardina, and L. Padgham. First principles planning in BDI systems. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)*, Budapest, Hungary, May 2009. ACM Press.

[20] Y. Lespérance, G. De Giacomo, and A. N. Ozgovde. A model of contingent planning for agent programming languages. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 477–484, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[21] S. M. Majercik and M. L. Littman. MAXPLAN: A new approach to probabilistic planning. In *Artificial Intelligence Planning Systems*, pages 86–93, 1998.

[22] A. U. Mallya and M. N. Huhns. Commitments among agents. *IEEE Internet Computing*, 7(4):90–93, 2003.

[23] A. U. Mallya and M. P. Singh. Modeling exceptions via commitment protocols. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 122–129, New York, NY, USA, 2005. ACM.

[24] F. R. Meneguzzi, A. F. Zorzo, and M. da Costa Móra. Propositional planning in bdi agents. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 58–63, New York, NY, USA, 2004. ACM.

[25] M. C. Mora, J. G. Lopes, R. M. Viccariz, and H. Coelho. Bdi models and systems: Reducing the gap. *Intelligent Agents V: Agents Theories, Architectures, and Languages*, 5:11–27, 1999.

[26] D. J. Musliner, E. H. Durfee, and K. G. Shin. CIRCA: A cooperative intelligent real time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1561–1574, - 1993.

[27] D. Nau, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. Shop2: An htn planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[28] Y. pa So and E. H. Durfee. Designing tree-structured organizations for computational agents. *Computational and Mathematical Organization Theory*, 2:219–246, 1996.

[29] S. Panagiotidi, J. Vazquez-Salceda, A.-N. S., O.-M. S., W. S., C. R., and S. P. Intelligent contracting agents language. In *Proceedings of Communication, Interaction and Social Intelligence (AISB 2008)*, Aberdeen, Scotland, Apr 2008.

[30] H. E. Pattison, D. D. Corkill, and V. R. Lesser. Instantiating Descriptions of Organizational Structures. In M. N. Huhns, editor, *Distributed Artificial Intelligence, Research Notes in Artificial Intelligence*, volume I, pages 59–96. Pitman Publishers, 1987.

[31] H. B. Planning, L. D. Silva, and L. Padgham. A comparison of bdi based real-time reasoning and htn based planning. In *In 17th Australian Joint Conference on Artificial Intelligence*, pages 1167–1173. Springer, 2004.

[32] E. Sacerdoti. *A Structure for Plans Behavior*. Elsevier, Amsterdam, 1977.

[33] T. Sandholm and V. Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework, 1999.

[34] S. Sardina, L. de Silva, and L. Padgham. Hierarchical planning in bdi agent programming languages: A formal approach. In *AAMAS '06: Proceedings of the 5th international joint conference on Autonomous agents and multiagent systems*, pages 1001–1008. ACM Press, 2006.

[35] S. Sardina and L. Padgham. Goals in the context of bdi plan failure and planning. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[36] L. D. Silva and L. Padgham. Planning on demand in bdi systems (poster). In *In Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*, 2005.

[37] M. P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7:97–113, 1999.

[38] M. P. Singh, A. S. Rao, and M. P. Georgeff. *Multiagent Systems A Modern Approach to Distributed Artificial Intelligence*, chapter Formal Methods in DAI: Logic Based Representation and Reasoning, pages 201–258. MIT Press, Cambridge, MA., 1999.

[39] A. Tate. Intelligible ai planning. In M. Bramer, A. Preece, and F. Coenen, editors, *Research and Development in Intelligent Systems XVII (Proc. 20th ES)*, pages 3–16. Springer, 2000.

[40] A. Tate. <I-N-C-A>: An ontology for mixed-initiative synthesis tasks. In *Proceedings of the Workshop on Mixed-Initiative Intelligent Systems (MIIS) at the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 125–130, Acapulco, Mexico, August 2003.

[41] J. Vokřínek, A. Komenda, and M. Pěchouček. Decommitting in multi-agent execution in non-deterministic environment: Experimental approach. In *AAMAS '09: Proceedings of the eight international joint conference on Autonomous agents and multiagent systems*, 2009.

[42] J. Vokřínek, A. Komenda, and M. Pěchouček. Relaxation of social commitments in multi-agent dynamic environment. In *Proceedings of International Conference on Agents and Artificial Intelligence (ICAART09)*. Springer, 19-21 January 2009.

[43] I. Wallace and M. Rovatsos. Bounded practical social reasoning in the esb framework. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 1097–1104, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[44] G. Wickler, A. Komenda, M. Pechoucek, A. Tate, and J. Vokrinek. Multi-agent planning with decommitment. In *KSCO '09: Knowledge Systems for Coalition Operations 2009*, Chilworth Manor, Southampton, UK, Mar-Apr 2009.

[45] G. Wickler, S. Potter, and A. Tate. Using i-x process panels as intelligent to-do lists for agent coordination in emergency response. *International Journal of Intelligent Control and Systems (IJICS), Special Issue on Emergency Management Systems*, 2006.

[46] G. Wickler, S. Potter, A. Tate, M. Pechoucek, and E. Semsch. Planning and choosing: Augmenting htn-based agents with mental attitudes. In *IAT '07: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 222–228, Washington, DC, USA, 2007. IEEE Computer Society.

[47] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative procedural goals in intelligent agent systems. In *In Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002*, pages 470–481, 2002.

[48] M. Wooldridge. *Reasoning about Rational Agents*. Intelligent robotics and autonomous agents. The MIT Press, 2000.

[49] M. Wooldridge and N. Jennings. Cooperative problem solving. *Journal of Logics and Computation*, 9(4):563–594, 1999.

[50] J. Xing and M. P. Singh. Formalization of commitment-based agent interaction. In *SAC '01: Proceedings of the 2001 ACM symposium on Applied computing*, pages 115–120, New York, NY, USA, 2001. ACM.

[51] P. Yolum and M. R. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. pages 527–534. ACM Press, 2002.