Specification and Synthesis of Plans Using the Features and Fluents Framework

Lars Karlsson
Email: larka@ida.liu.se

Abstract

An autonomous agent operating in a dynamical environment will face a number of different reasoning problems, one of which is how to plan its actions in order to pursue its goals. For this purpose, it is important that the agent represents its knowledge about the world in a coherent, expressive and well-understood way, in our case the temporal logics from Erik Sandewall's "Features and Fluents" framework. However, most existing planning systems make no use of temporal logics, but have specialised representations such as the Strips formalism and hierarchical task networks. In order to benefit from the techniques used by these planners, it is useful to analyse and reconstruct them within the given framework. This includes making explicit the ontological and epistemological assumptions underlying the planners; representing plans as entities of the temporal logic; and reconstructing the algorithms in terms of the new representation. The two planners Strips and Tweak have been analysed and reconstructed in this way.

This work has been supported by the Swedish National Board for Industrial Development (NUTEK grant 5222-92-5615) as part of the ESPRIT Basic Research Action No 6156-Drums II.

IDA Technical Report 1994
LiTH-IDA-R-94-28
ISSN-0281-4250

Department of Computer
and Information Science,
Linköping University,
S-581 83 Linköping, Sweden

About this report

This report is a summary and a selection from a forth-coming Master's thesis. The proofs of the theorems appearing here can be found in the thesis.

1 Introduction

The construction of artificial devices, such as robots, that can reason about and operate in a complex and changing environment has been one of the major goals for artificial intelligence since the emergence of the field in the 50's. Such devices are commonly referred to as autonomous agents.

An agent applies symbolic computations to draw conclusions and develop a rational behavior[1]. This involves modeling the world in some kind of symbolic language, for instance a programming language or a language of logic. Furthermore, it involves representing and computing the problems of reasoning the agent will face in its environment. Two such problems which have been the subjects of extensive studies are:

- reasoning about how actions affect the world and how the world changes over time;
- finding plans consisting of actions which lead to some desired goals.

The former problem is commonly referred to as *reasoning about action and change* or *temporal reasoning*; the latter most commonly as *planning*, or more specifically as *plan construction*, *plan synthesis* or *plan generation*.

1.1 Temporal Reasoning and Planning

The problems of temporal reasoning and planning have long histories in AI. As they are two closely related topics, their respective histories have much in common and the influence has been mutual.

The first temporal representation used in AI was the situation calculus, introduced by McCarthy and Hayes [McC62, MH69] and based on first-order logic. Green [Gre69] and Kowalski [Kow77] both used a situation calculus-type of representation, and showed how theorem proving can be used to find plans which solve specific problems. Other examples of temporal logics applied to planning are Pednault's *Action Description Language* (ADL) [Ped86] and Allen's logic of temporal intervals [All84, All91]. Rosenschein [Ros81] used dynamic logic for representing plans.

However, the main part of planning systems do not use temporal logics, especially if we consider planners that have actually been implemented and applied for solving realistic problems. These planners generally follow an operator-based approach and fall under two main categories: *precondition-achievement* planners and *hierarchical* planners [Tat77, CT91, Wil88].

[1] There are alternative approaches where symbolic reasoning is not taken as a primitive aspect of intelligence, e.g. connectionist models [Gro88] and Brook's subsumption architecture [Bro91].

The precondition-achievement planner STRIPS by Fikes and Nilsson [FN71] represents situations as sets of first-order logic sentences. In order to sidestep the problem with frame axioms in situation calculus, STRIPS does not represent actions (called operators) and their effects as entities within the logic, but as transformations on sets of sentences. Some old sentences are deleted, and some new sentences are added. This makes it easy to compute what is true or false in a situation, but limits the expressiveness considerably. For instance, the STRIPS formalism cannot represent context-dependent actions.

1.2 Features and Fluents

This thesis follows the tradition of using temporal logics for planning. The subject is treated within Sandewall's *Features and Fluents* framework [San94, SS94], which takes a systematic approach to the analysis and construction of temporal logics. Sandewall's methodology is based on five principles:

- The use of an *underlying semantics* which characterizes systems with agents and change; these systems are called *dynamical inhabited systems*.
- The use of the underlying semantics for a formal definition of what are the *intended models* of logics for the representation of such systems.
- The use of an *ontological taxonomy* which is strictly defined in terms of the underlying semantics, and which characterizes classes of dynamical systems with specific restrictions on them.
- Explicit statement of *epistemological assumptions*.
- Formally proven assessments of the range of applicability of various (non-monotonic) entailment criterias, based on the strict definition on intended models.

The proposed logics use an integer time structure, which can either be linear, or combined with event-based branching in a manner similar to situation calculus. Higher-level reasoning problems such as planning can be defined in terms of some basic operations such as temporal projection.

1.3 Objectives and Motivations

The topic of this report is on the formalization of the planning problem and on the analysis and reconstruction of existing planners and the ways they model a changing world, all within the *Features and Fluents* framework. The emphasis is on representation and on basic operations on this representation; the algorithms used by these planners need not be changed in any relevant aspects. Precondition-achievement operator planning is studied in this way. This study is a prerequisite for the analysis of more advanced planners, and for the development of planners suitable for an agent operating in real-time in a realistically complex environment.

1.3.1 Temporal Logics

Why bother about temporal logics, when most existing planners seem to do well without them? One motivation concerns integration. Planning is but one of the reasoning capabilities required in a dynamic and complex environment. This puts high demands of expressiveness and flexibility on whatever representation is being used. Hierarchical task networks [Tat77, CT91] might be good for hierarchical planning, but we are less convinced they are good for e.g. anticipation (fore-seeing the future) and learning.

A second motivation is the implicit assumptions about the properties of the world which underly these planners. They might eventually turn out to be too restrictive to meet the requirements of modeling a complex and continuously changing environment. The more advanced properties the world has, the more important it is to have a good and solid model of time and change to build on.

Finally, we hope that studying existing planners from a temporal logics perspective will contribute to a better understanding of how current planners work, especially concerning how they represent the world.

1.3.2 Planning and the Systematic Approach

Our investigation will be based on Sandewall's systematic approach. Sandewall states that [San94, p.32]

> ...The ego's[2] understanding of its world is in terms of (discrete valued) features and occurrences. The understanding is intended to be used for prediction, postdiction (e.g. diagnosis), generalization, planning etc. In order to realize this understanding process in a correct and efficient fashion, it is useful to identify special cases with various restrictions on the structure of the world, over the general framework which has been defined so far.

This is a highly valid statement when it comes to planning. When doing plan synthesis, the ego will continuously modify its picture of the (intended) future development of the world by adding action instances and constraints. We strongly believe that in order to do efficient planning, efficient reasoning about action and change is necessary. Restrictions on the world structure can therefore result in considerable gains in planning efficiency.

When discussing efficiency, we should observe that planning with a first-order representation with variables or function symbols (e.g. STRIPS) generally is undecidable [Cha87, ENS92]. Consequently, standard complexity theory does not suffice when it comes to assessing the difficulty of non-trivial planning problems and techniques. However, measures from complexity theory can be applied to subproblems such as temporal projection.

[2] The ego is the integrated higher-level reasoning functions of the agent.

2 Plan Specification

Scenario descriptions are the tools for specifying the agent's knowledge, plans and goals. A scenario description typically contains information about action laws, action occurrences and observations of the state in the world. We shall illustrate this with an example. Assume the agent has the following knowledge:

law1	if one enters a restaurant, one will be inside the restaurant.	
law2	if one is inside a restaurant and one dines, one will no longer be hungry.	(1)
obs1	At time 0, I am outside the restaurant, and I am hungry.	

The agent also has a goal:

goal1 I want to stop being hungry. (2)

We shall use the term *goal description* for a scenario description expressing a (set of) goal(s).

Planning is, given a (set of) scenario description(s) of the current state of the world and a goal description, to find a set of action instances that, when applied to the scenario descriptions, describes a set of developments that meets the requirements of the goal description.

Pursuing our example, a plan which achieves the goal is:

scd1	I enter the restaurant between 2 and 4.	(3)
scd2	I dine between 5 and 7.	

Clearly, this plan achieves the goal of not being hungry.

Usually some formal language is used in the scenario descriptions, for instance the fluent logic which will be defined in this section. How the scenario descriptions are constituted and what conclusions are valid depends on the *ontology* of the world and the *epistemological* assumptions under which the agent operate.

2.1 The Ontological Taxonomy

We stated in the introductory chapter that is our belief that restrictions on the world structure often have considerable impact on the difficulty of temporal reasoning and planning problems. As a tool for the analysis and specification of such restrictions, Sandewall provides a *taxonomy of ontological specialities and subspecialities*, characterized by code letters. Some of these specialities are presented in fig 1 and fig 2.

The letters can be combined into *ontological designators.* For instance, the ontological designator **IAdC** characterizes the assumption of strong inertia, alternative but deterministic results of actions and possible concurrency of actions.

Let us see how the ontological taxonomy relates to traditional planning. The most well-known formalism is the STRIPS formalism [FN71], where actions are characterized by one set of preconditions and one set of effects,

I	Strong inertia. Integer time.
A	Alternative results of actions (context dependency and nondeterminism).
U	Local ramification - dependencies between features of the objects directly involved in an action.
D	Structural ramification - dependencies with features of objects that are only indirectly involved in an action.
C	Concurrency of actions.

Figure 1: Some ontological specialities.

Is	Single-step actions. Implies **Ie**.
Ie	Encapsulated actions, i.e. they are only specified in terms of pre- and postconditions.
Ib	Binary features (T, F).
Ad	Deterministic actions.

Figure 2: Some ontological subspecialities for **I** and **A**.

independent of state. (Actually the effects are defined in terms of *two* sets, called the add and delete lists. STRIPS will be given a more detailed investigation later in this paper.) STRIPS appears to fall within the **Isb** assumption. Some recent planners such as SIPE [Wil88] deals with concurrency in terms of resource sharing, and would consequently belong to the **IebC** assumption. Pednault's ADL supports planning with context-dependent actions (**IAd**) [Ped88].

2.2 The Epistemological Taxonomy

The agent might have more or less correct and complete knowledge about action laws, action and event occurrences and the state of the world. This knowledge is usually expressed in scenario descriptions. Sandewall provides a taxonomy of epistemological specialities (fig 3) and properties (fig 4) which are applicable to scenario descriptions.

$\mathcal{N}$	No epistemological assumption.
$\mathcal{K}$	Correct and accurate knowledge.
$\mathcal{Q}$	Qualified action laws. Effects of actions may be unspecified when preconditions are false.
$\mathcal{U}_\mathsf{a}$	Uninformed about actions. Some instances of actions remain unspecified.
$\mathcal{U}_\mathsf{e}$	Uninformed about events. Some instances of events remain unspecified.
$\mathcal{U}_\mathsf{l}$	Uninformed about actions laws. Don't always know the effects of actions, or even what features they affect.

Figure 3: Some epistemological specialities.

From the perspective of traditional planning, the specialities $\mathcal{K}_\mathsf{p}$ and $\mathcal{Q}_\mathsf{p}$ are of most interest. $\mathcal{K}$ and $\mathcal{Q}$ both stands for scenarios where the agent knows about anything that has happened and/or will happen. This the case

$\mathcal{K}\mathsf{p}$	Correct and accurate knowledge of the initial state, and no information of any later state.
$\mathcal{K}\mathsf{g}$	Correct and accurate knowledge, only "ground" scenarios (no variables).

Figure 4: Some epistemological properties for $\mathcal{K}$.

when the agent is alone in the world and no external events can interfere; an assumption commonly made in planning. The letter p corresponds to the restriction of information to the initial situation which is common in planning.

2.3 Formal Definition of Reasoning Problems

Definition 1 *A* **scenario description** [San94] *is a tuple*

$$\langle \mathcal{Z}, \Gamma_1, \Gamma_2, \ldots, \Gamma_k \rangle$$

where $\mathcal{Z}$ identifies the epistemological assumptions made about the scenario, and the succeeding elements are sets of formulae or other formal objects. The number and types of the succeeding elements are specific for each choice of $\mathcal{Z}$.

For the epistemological assumption $\mathcal{K}$, a scenario description — called a chronicle — is a tuple

$$\langle \mathcal{K}, \mathcal{O}, \mathsf{Diw}, \textsc{Scd}, \textsc{Obs} \rangle$$

where $\mathcal{O}$ is an object domain; $\mathsf{Diw} = \langle \Psi, \textsc{Law} \rangle$ is a world description where Ψ is an ontological designator and $\textsc{Law}$ is a set of action laws; $\textsc{Scd}$ is a schedule specifying a number of action occurrences and their temporal order; and $\textsc{Obs}$ is a set of observations of feature values.

For the epistemological assumption $\mathcal{Q}$, the action laws are split into two parts. The action qualification laws state the conditions for when the result of an action is defined. The action effect laws then define the result of the action, provided that the qualifications of the action are true.

A scenario description (chronicle) for $\mathcal{Q}$ is a tuple

$$\langle \mathcal{Q}, \mathcal{O}, \mathsf{Diw}, \textsc{Scd}, \textsc{Obs} \rangle$$

where $\mathcal{O}$ is an object domain; $\mathsf{Diw} = \langle \Psi, \textsc{Qlaw}, \textsc{Elaw} \rangle$ is a world description where Ψ is an ontological designator, $\textsc{Qlaw}$ is a set of action qualification laws and $\textsc{Elaw}$ is a set of action effect laws; $\textsc{Scd}$ is a schedule specifying a number of action occurrences and their temporal order; and $\textsc{Obs}$ is a set of observations of feature values.

Let $\Vdash$ be an **entailment relation** for scenario descriptions. Sandewall [San94] defines this relation in terms of the *underlying semantics*, a topic we will not go into in this report.

Definition 2 *A* **k-adic reasoning problem** [San94] *is a fourtuple*

$$\langle\langle\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_k\rangle, \Xi, \leadsto, \ll\rangle$$

where all Υ_i are scenario descriptions, Ξ is a set of scenario descriptions wherein the solution is to be found, $\leadsto$ is a (k+1)-ary relation over scenario descriptions written as

$$\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_k \leadsto \Upsilon,$$

and $\ll$ is a preference relation over Ξ. A scenario $\Upsilon' \in \Xi$ is a **solution** *to this reasoning problem iff*

$$\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_k \leadsto \Upsilon',$$

and Υ' is $\ll$-minimal among those $y \in \Xi$ for which $\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_k \leadsto y$.

Basic plan synthesis is the dyadic reasoning problem

$$\langle\langle\Upsilon_W, \Upsilon_G\rangle, \Xi, \leadsto, \ll\rangle$$

where Υ_W is a scenario description representing the world from now to some point in the future, Υ_G is a scenario description specifying the goals that are to be achieved, Ξ is a set of scenarios representing potential plans, and $\ll$ is some optimality criterion, e.g. plan length. Then

$$\Upsilon_W, \Upsilon_G \leadsto \Upsilon'$$

iff

$$(\Upsilon_W \cup \Upsilon') \Vdash \Upsilon_G.$$

and Υ' is $\ll$-minimial in Ξ. The expression $\Upsilon_1 \cup \Upsilon_2$ denotes the scenario description with the unions of those elements which are sets; all elements that are not sets should be identical for the two scenario descriptions involved.

Basic plan synthesis in $\mathcal{K}$ can be described as a reasoning problem where

$$\langle\mathcal{K}, \mathcal{O}, \mathsf{Diw}, \emptyset, \textsc{Obs}\rangle, \langle\mathcal{N}, \mathcal{O}, \textsc{Goal}\rangle \leadsto \langle\mathcal{K}, \mathcal{O}, \emptyset, \textsc{Scd}, \emptyset\rangle$$

iff

$$\langle\mathcal{K}, \mathcal{O}, \mathsf{Diw}, \textsc{Scd}, \textsc{Obs}\rangle \Vdash \langle\mathcal{N}, \mathcal{O}, \textsc{Goal}\rangle$$

where Ξ is the set of all scenario descriptions $\langle\mathcal{K}, \mathcal{O}, \emptyset, \textsc{Scd}, \emptyset\rangle$ containing only actions that the agent is allowed to perform.

Similarly, basic plan synthesis in $\mathcal{Q}$ can be described as a reasoning problem where

$$\langle\mathcal{Q}, \mathcal{O}, \mathsf{Diw}, \emptyset, \textsc{Obs}\rangle, \langle\mathcal{N}, \mathcal{O}, \textsc{Goal}\rangle \leadsto \langle\mathcal{Q}, \mathcal{O}, \emptyset, \textsc{Scd}, \emptyset\rangle$$

iff

$$\langle\mathcal{Q}, \mathcal{O}, \mathsf{Diw}, \textsc{Scd}, \textsc{Obs}\rangle \Vdash \langle\mathcal{N}, \mathcal{O}, \textsc{Goal}\rangle.$$

2.4 A Language for Scenario Descriptions

In the beginning of this report, we pointed to the advantages of logics for plan representation. Here, we will define a reified and typed *fluent logic* (FL) with the non-negative integers for time domain. It is an extension of a fluent logic by Doherty and Lukaszewicz [DL94, Doh94], which in turn is based on Sandewall's logics. The extension lies in the introduction of fluents with arguments.

FL is a representation which is not restricted to plan synthesis. That is one of the points with selecting it; it gives us a reasonable chance to construct plan synthesisers with a relatively high level of integration when it comes to knowledge sharing between reasoning functions.

2.4.1 The Base Language $\mathcal{L}(FL)$

The fluent logic is a sorted logic with equality. There are three sorts: $\mathbf{t}$ for time-points, $\mathbf{f}$ for fluents and $\mathbf{e}$ for atemporal objects. A vocabulary Λ is a tuple $\langle C_t, C_f, C_e, V_t, V_f, V_e, F_t, R, d\rangle$ where d is a sort function and

1. C_t is an enumerable set of temporal constants; $d(t) = \mathbf{t}$. Here, the integers will be used.

2. C_f is an enumerable set of fluent function constants; $d(f_i) = \langle \mathbf{e}^{m_i} \rightarrow \mathbf{f}\rangle$.

3. C_e is an enumerable set of atemporal constants; $d(e) = \mathbf{e}$.

4. V_t is an enumerable set of temporal variables. Here, s, t will be used, possibly primed or subscribed.

5. V_f is an enumerable set of fluent variables. Here, f, g will be used, possibly primed or subscribed.

6. V_e is an enumerable set of atemporal variables. Here, x, y will be used, possibly primed or subscribed.

7. F_t is a set of temporal functions; $d(t_i) = \langle \mathbf{t}^{n_i} \rightarrow \mathbf{t}\rangle$.

8. $R = \{Holds, Occluded, <, =_t, =_f, =_e\}$ is a set of relations, where $d(Holds) = \langle \mathbf{t} \times \mathbf{f}\rangle$ and $d(Occluded) = \langle \mathbf{t} \times \mathbf{f}\rangle$ are predicates for fluent values and fluent changes, $d(<) = \langle \mathbf{t} \times \mathbf{t}\rangle$ is a temporal order, and finally $d(=_t) = \langle \mathbf{t} \times \mathbf{t}\rangle$, $d(=_f) = \langle \mathbf{f} \times \mathbf{f}\rangle$ and $d(=_e) = \langle \mathbf{e} \times \mathbf{e}\rangle$ are equality predicates. Henceforth, only $=$ will be used when the context tells what sort it is.

The usual set of connectives and quantifiers $\wedge, \vee, \neg, \Rightarrow, \forall, \exists$ is provided.

2.4.2 The Surface Language $\mathcal{L}(SD)$

In order to obtain a more compact representation, a surface language is defined on top of the base language. For the surface language, we introduce action statements on the form

$$[\boldsymbol{s}, \boldsymbol{t}]\, A(\overline{\boldsymbol{e}})$$

and action expansion laws on the form:

$$[s, t]\, A(\overline{x}) \rightsquigarrow [s, t]\, \alpha(\overline{x})$$

where $[s, t]\, \alpha(\overline{x})$ is a formula describing the preconditions, effects and constraints of the action A.

A number of abbreviations are also introduced. Let δ denote an arbitrary fluent expression $f(\overline{e})$.

$$[\boldsymbol{t}]\, \delta \stackrel{\text{def}}{=} Holds(\boldsymbol{t}, \delta) \tag{4}$$

$$[\boldsymbol{s}, \boldsymbol{t}]\, \delta \stackrel{\text{def}}{=} \forall t. \boldsymbol{s} \leq t \leq \boldsymbol{t} \Rightarrow Holds(t, \delta) \tag{5}$$

$$[\boldsymbol{t}]\, \delta^* \stackrel{\text{def}}{=} Occlude(\boldsymbol{t}, \delta) \tag{6}$$

$$[\boldsymbol{s}, \boldsymbol{t}]\, \delta^* \stackrel{\text{def}}{=} \forall t. \boldsymbol{s} \leq t \leq \boldsymbol{t} \Rightarrow Occlude(t, \delta) \tag{7}$$

$$\begin{aligned}[\boldsymbol{s}, \boldsymbol{t}]\, \delta{:=}T \stackrel{\text{def}}{=}\ & \exists t. (\boldsymbol{s} \leq t < \boldsymbol{t} \wedge \forall t'. (t < t' \leq \boldsymbol{t} \Rightarrow Holds(t', \delta))) \\ & \wedge \forall t. (\boldsymbol{s} < t \leq \boldsymbol{t} \Rightarrow Occlude(t, \delta))\end{aligned} \tag{8}$$

$$\begin{aligned}[\boldsymbol{s}, \boldsymbol{t}]\, \delta{:=}F \stackrel{\text{def}}{=}\ & \exists t. (\boldsymbol{s} \leq t < \boldsymbol{t} \wedge \forall t'. (t < t' \leq \boldsymbol{t} \Rightarrow \neg Holds(t', \delta))) \\ & \wedge \forall t. (\boldsymbol{s} < t \leq \boldsymbol{t} \Rightarrow Occlude(t, \delta))\end{aligned} \tag{9}$$

$$[\boldsymbol{t}]\, \neg\delta \stackrel{\text{def}}{=} \neg [\boldsymbol{t}]\, \delta \tag{10}$$

The translation from $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$ is a two-step process. First, all action statements are expanded according to the action expansion laws. For instance, if

$$[s, t]\, Fire(x, y) \rightsquigarrow [s]\, loaded(x) \Rightarrow [s, t]\, lead(y){:=}T$$

is an action expansion law, then $[2, 4]\, Fire(Rifle, Fred)$ will expand to

$$[2]\, loaded(Rifle) \Rightarrow [2, 4]\, dead(Fred){:=}T.$$

Second, the abbreviations are used to translate the expanded scenario description to $\mathcal{L}(SD)$.

2.4.3 Occlusion and Reassignment

The purpose of the *Occlude* predicate is to explicitly suspend the inertia assumption. Inertia implies the minimization of changes in fluents in a scenario description. However, if $Occlude(\boldsymbol{t}, \delta)$ is among the premises, then models where δ changes its truth value ($Holds(\boldsymbol{t} - 1, \delta) \wedge \neg Holds(\boldsymbol{t}, \delta)$ or vice versa) and models where δ retains its truth value will be equally preferred. *Reassignment* ($[\boldsymbol{s}, \boldsymbol{t}]\, \delta{:=}T$) is used to express change to a new value within an interval.

2.4.4 Chronicles

A *chronicle description* in FL is a $\mathcal{K}$-**IA** scenario description

$$\Gamma = \langle \mathcal{K}, \mathcal{O}, \textsc{Law}, \textsc{Scd}, \textsc{Obs} \rangle$$

where

$\mathcal{O}$ is an object domain description.

$\textsc{Law}$ is a set of action expansion laws

$\textsc{Scd}$ is a schedule; a set of action statements and temporal and atemporal constraints.

$\textsc{Obs}$ is an observation set; a set of formulae of the form $[\boldsymbol{t}]\,\delta$ or $[\boldsymbol{t}, \boldsymbol{t}']\,\delta$.

We write $\textsc{Law}(\textsc{Scd})$ to denote the expansion of the set $\textsc{Scd}$ according to the laws in $\textsc{Law}$. Anything not an action instance will pass unchanged.

For convenience, Γ_O, Γ_{LAW}, Γ_{SCD} and Γ_{OBS} will be used to denote the components of a chronicle Γ. The $\mathcal{K}$ can be omitted when it is clear from context. When describing a scenario, we will use the labels law, scd and obs to distinguish between expressions belonging to $\textsc{Law}$, $\textsc{Scd}$ and $\textsc{Obs}$. The classical Yale Shooting Scenario can be expressed as below, in unexpanded version.

$$\begin{array}{ll} \mathsf{law1} & [s,t]\, Load(x) \leadsto [s,t]\, loaded(y){:=}T \\ \mathsf{law2} & [s,t]\, Fire(x,y) \leadsto [s]\, loaded(x) \Rightarrow [s,t]\, alive(y){:=}F \\ \mathsf{obs1} & [0]\, alive(Fred) \wedge \neg loaded(Rifle) \\ \mathsf{scd1} & [2,4]\, Load(Rifle) \\ \mathsf{scd2} & [5,6]\, Fire(Rifle, Fred) \end{array} \tag{11}$$

In expanded version, it reads as follows.

$$\begin{array}{ll} \mathsf{obs1} & [0]\, alive(Fred) \wedge \neg loaded(Rifle) \\ \mathsf{scd1} & [2,4]\, loaded(Rifle){:=}T \\ \mathsf{scd2} & [5]\, loaded(Rifle) \Rightarrow [5,6]\, alive(Fred){:=}F \end{array} \tag{12}$$

Observe that the elements labeled scd now are members of $\textsc{Law}(\textsc{Scd})$. This scenario can then easily be reduced to the base language.

2.4.5 Object and Fluent Domains

The object domain consists of a set of distinct objects. A subset of the atemporal constants C_e is partitioned off to specifically denote these object: such constants will be called *fixed atemporal constants*. The remaining constants will serve as "global variables"; these will be called *unfixed atemporal constants*. We will use capital V with index for these.

When talking about unspecified atemporal constants, we will use the following boldface letters to distinguish between constants of different functionality.

- $\boldsymbol{c}$ for fixed atemporal constants.

- $\boldsymbol{v}$ for unfixed atemporal constants.
- $\boldsymbol{e}$ for atemporal constants of any of the two classes above.

A unique name assumption should be imposed on the fixed constants.

$$\Gamma_{UNA} = \bigcup_{1 \le i < j \le n} \{\boldsymbol{c}_i \neq \boldsymbol{c}_j\} \tag{13}$$

Sandewall's logics uses a finite object domain. We shall leave this optional, as it does not appear to have any effect on the correctness of the proposed logics. A finite domain can be expressed with a domain closure axiom.

$$\Gamma_{DC} = \{\forall x.[x = \boldsymbol{c}_1 \vee \ldots \vee x = \boldsymbol{c}_n]\} \tag{14}$$

These sets of axioms constitute the object domain $\mathcal{O}$ of a chronicle description.

We shall use a set of unique fluent assumptions.

$$\Gamma_{UFA} \quad = \quad \bigcup_{i<j}\{\ \forall \overline{x}, \overline{y}.\boldsymbol{f}_i(\overline{x}) \neq \boldsymbol{f}_j(\overline{y})\} \tag{15}$$

The boldface letter $\boldsymbol{f}$ represents an unspecified fluent symbol.

Finally, we combine these sets into a set of domain assumptions, either with domain closure

$$\Gamma_{DAC} = \Gamma_{UNA} \cup \Gamma_{DC} \cup \Gamma_{UFA} \tag{16}$$

or without domain closure

$$\Gamma_{DA} = \Gamma_{UNA} \cup \Gamma_{UFA} \tag{17}$$

2.4.6 Time domain

The time domain will be the non-negative integers, denoted $0, 1, 2, \ldots$, with addition but without multiplication. A theory of this domain, including $<$ and the successor function $'$, is decidable as proven by Presburger (see e.g. [BJ80, p. 219]).

Besides the integers, which are fixed temporal constants, a set of unfixed temporal constants is available. The capital letter S with index is used to denote these.

2.4.7 Minimizing change in FL

The criterion used for minimization of change is called PMON (Pointwise Minimization of Occlusion with No-Change premises). PMON has been proven correct for the $\mathcal{K}$-**IA** class of scenarios [San94].

The idea is to minimize occlusion — "allowed change" — and then filter away those models where there are unoccluded changes. For this purpose, we use an additional axiom set of *nochange premises* as follows ($\oplus$ denotes 'exclusive or').

$$\Gamma_{NCH} = \{\ \forall f, t.Holds(t, f) \oplus Holds(t+1, f) \Rightarrow Occlude(t+1, f)\ \} \tag{18}$$

The PMON minimization policy is defined as follows, using second-order circumscription [Lif85, Luk90] to minimize *Occlude* while keeping *Holds* fixed. The circumscription operation is performed on scenario descriptions in the base language. We will use the abbreviation $U < V$ for $\forall \overline{x}.(U(\overline{x}) \Rightarrow V(\overline{x})) \wedge \neg(U(\overline{x}) \Rightarrow V(\overline{x}))$. Let $\Gamma(Occlude) = \textsc{Law}(\textsc{Scd})$ where $\textsc{Law}$ and $\textsc{Scd}$ are components of Γ. Then

$$PMON(\Gamma) = \\ Circ_{SO}(\Gamma(Occlude); Occlude; ()) \wedge \Gamma_{OBS} \wedge \Gamma_{NCH} \wedge \Gamma_{DA(C)} \quad (19)$$

where

$$Circ_{SO}(\Gamma(Occlude); Occlude; ()) = \\ \Gamma(Occlude) \wedge \forall \Phi.\neg[\Gamma(\Phi) \wedge \Gamma < Occlude] \quad (20)$$

We shall write

$$\Gamma \mid\!\sim \alpha \quad (21)$$

for $PMON(\Gamma) \vdash \alpha$, and

$$MOD^{PMON}(\Gamma) \quad (22)$$

for $MOD(PMON(\Gamma))$ where $MOD(T)$ denotes the model set of a theory T.

2.4.8 FL Chronicles as Logic Programs

Doherty [Doh94] has shown that PMON circumscription results in first-order-reducible theories. Consequently, classical theorem proving techniques can be applied to scenario descriptions.

2.4.9 Restricted Monotonicity of FL Scenario Descriptions

As shown by Doherty [Doh94], FL chronicles have the *restricted monotonicity property* [Lif93]. Some classes of statements of a non-monotonic formalism monotonically increases the set of valid conclusions when added to a set of premises. Such classes can be identified for FL scenario descriptions.

Definition 3 *A chronicle* $\Gamma' = \langle \mathcal{O}, \textsc{Law}, \textsc{Scd}', \textsc{Obs}' \rangle$ *is a* **monotonic extension under PMON** *of a chronicle* $\Gamma = \langle \mathcal{O}, \textsc{Law}, \textsc{Scd}, \textsc{Obs} \rangle$ *iff*

1. $\textsc{Obs} \subseteq \textsc{Obs}'$
2. $\textsc{Scd} \subseteq \textsc{Scd}'$
3. $MOD^{PMON}(\Gamma) \supseteq MOD^{PMON}(\Gamma')$.

Theorem 1 *A chronicle* $\Gamma' = \langle \mathcal{O}, \textsc{Law}, \textsc{Scd} \cup \gamma, \textsc{Obs} \rangle$ *is a monotonic extension of* $\Gamma = \langle \mathcal{O}, \textsc{Law}, \textsc{Scd}, \textsc{Obs} \rangle$ *if* γ *only contains elements on the forms* $\boldsymbol{e}_1 = \boldsymbol{e}_2$, $\boldsymbol{e}_1 \neq \boldsymbol{e}_2$, $\boldsymbol{t}_1 = \boldsymbol{t}_2$, $\boldsymbol{t}_1 \leq \boldsymbol{t}_2$, *and* $\boldsymbol{t}_1 < \boldsymbol{t}_2$.

A chronicle $\Gamma'' = \langle \mathcal{O}, \textsc{Law}, \textsc{Scd}, \textsc{Obs} \cup \gamma \rangle$ *is a monotonic extension of* Γ *for any set of observations* γ.

Proof
See Doherty [Doh94].

To sum it up, temporal and atemporal constraints and observations can be added without invalidating any previous conclusions.

2.4.10 FL for $\mathcal{Q}$

No logic for the epistemological speciality $\mathcal{Q}$ is presented by Sandewall [San94], and none will be proposed here. As soon will be shown, this problem can be side-stepped by reasoning in $\mathcal{K}$.

2.5 Plan Specification

Our intention is to use scenario descriptions to specify the world of a planning problem, the goals, and the resulting plan. We shall illustrate with the restaurant example under a $\mathcal{K}$-**IA** assumption. The agent operating in the IDS has some knowledge about action laws and the state of the world.

$$\begin{array}{ll}
\mathsf{law1} & [t,t']\,EnterRestaurant \leadsto [t]\,inStreet \Rightarrow \\
 & \quad [t,t']\,inStreet{:=}F \wedge [t,t']\,inRest{:=}T \\
\mathsf{law2} & [t,t']\,Dine \leadsto [t]\,inRest \Rightarrow [t,t']\,hungry{:=}F \\
\mathsf{obs1} & [0]\,hungry \\
\mathsf{obs2} & [0]\,inStreet \\
\mathsf{obs3} & [0]\,\neg inRest
\end{array} \tag{23}$$

The goal of the agent is to stop being hungry; we specify the time-point to 10.

$$\mathsf{goal1} \quad [10]\,\neg hungry \tag{24}$$

The agent forms a plan to enter the restaurant and then dine.

$$\begin{array}{ll}
\mathsf{scd1} & [2,5]\,EnterRestaurant \\
\mathsf{scd2} & [7,9]\,Dine
\end{array} \tag{25}$$

Let Γ denote the scenario description obtained by applying the plan to the initial state. The PMON *completion* [San94] of Γ is as follows:

$$\begin{array}{l}
[0,2]\,hungry \wedge inStreet \wedge \neg inRest \\
[3,4]\,hungry \\
[5,7]\,hungry \wedge \neg inStreet \wedge inRest \\
[8]\,\neg inStreet \wedge inRest \\
[9,\infty)\,\neg hungry \wedge \neg inStreet \wedge inRest
\end{array} \tag{26}$$

The goal $[10]\,\neg hungry$ is clearly achieved by the plan.

2.6 Reasoning about Partial Plans

There might be stages in the plan synthesis process where the ego has to reason about partial plans where unexecutable actions occur. This is for instance the case with regressive plan synthesisers, where the last actions of the plan are added first. In general, the plan is not executable, or valid, until a complete solution to the planning problem has been found. We shall call plans occurring in the intermediate stages of the plan synthesis process *partial plans*.

Consider the following example.

$$\begin{array}{ll} \textsf{law1} & [t,t']\,EnterRestaurant \leadsto [t]\,inStreet \Rightarrow \\ & \quad [t,t']\,inStreet{:=}F \wedge [t,t']\,inRest{:=}T \\ \textsf{law2} & [t,t']\,Dine \leadsto [t]\,inRest \Rightarrow [t,t']\,hungry{:=}F \\ \textsf{obs1} & [0]\,hungry \\ \textsf{obs2} & [0]\,inStreet \\ \textsf{obs3} & [0]\,\neg inRest \\ \textsf{scd1} & [7,9]\,Dine \\ \textsf{goal1} & [10]\,\neg hungry \end{array} \tag{27}$$

In this partial plan for satisfying the hunger of the agent, one action has been added to the plan: $[7,9]\,Dine$. In this situation, we might want to reason about how adding this action affects the world. However, no new conclusions can be obtained by reasoning in the epistemological speciality $\mathcal{K}$. The precondition $[7]\,inRest$ of $[7,9]\,Dine$ is not true, and therefore the action has no effects. Generally, in $\mathcal{K}$ a partial plan might be totally without any effects, or only a few of the intended effects occur.

Turning to the epistemological speciality $\mathcal{Q}$, a similar problem can be observed.

$$\begin{array}{ll} \textsf{qlaw1} & [t,t']\,EnterRestaurant \leadsto [t]\,inStreet \\ \textsf{elaw1} & [t,t']\,EnterRestaurant \leadsto \\ & \quad [t,t']\,inStreet{:=}F \wedge [t,t']\,inRest{:=}T \\ \textsf{qlaw2} & [t,t']\,Dine \leadsto [t]\,inRest \\ \textsf{elaw2} & [t,t']\,Dine \leadsto [t,t']\,hungry{:=}F \\ \textsf{obs1} & [0]\,hungry \\ \textsf{obs2} & [0]\,inStreet \\ \textsf{obs3} & [0]\,\neg inRest \\ \textsf{scd1} & [7,9]\,Dine \\ \textsf{goal1} & [10]\,\neg hungry \end{array} \tag{28}$$

In $\mathcal{Q}$, an unexecutable action results in the undefined state. In either case, we might be in trouble when it comes to drawing interesting conclusions from a partial plan.

There is nothing wrong about the presence of unexecutable actions in a partial plan. Plans consist of the agent's *intentions* to perform actions, and not of actual action occurrences in the world. Furthermore, the intended actions have intended effects from which the agent should be able to draw conclusions. Consequently, the agent should be able to reason about a partial plan *as if* it were valid. The rest of this chapter is dedicated to solving this problem.

There is an interesting relation between basic plan synthesis in $\mathcal{K}$ and $\mathcal{Q}$ which we can take advantage of.

Definition 4 *A set of action laws* $\textsc{Law}$ *is* **equivalent** *to an action effect law set* $\textsc{Elaw}$ *and an action qualification law set* $\textsc{Qlaw}$ *if for each action* E *and each state* r*,*

- *if the state* r *doesn't satisfy* $\textsc{Qlaw}(E)$*, then* $\textsc{Law}(E)$ *states that* E *has no effects when executed in* r*, and*
- *if the state* r *satisfies* $\textsc{Qlaw}(E)$*, then* $\textsc{Law}(E)$ *and* $\textsc{Elaw}(E)$ *states identical effects of* E *when executed in* r*.*

Theorem 2 *Let* $\textsc{Qlaw}$*,* $\textsc{Elaw}$ *be a pair of action qualification and and effect law sets for a* $\mathcal{Q}$*-world and let* $\textsc{Law}$ *be an action law set equivalent to this pair. Let* $\textsc{True}$ *map all actions to the value* T *for true.* Ψ *is* **IA***. If*

$$\langle \mathcal{K}, \mathcal{O}, \langle \Psi, \textsc{Elaw} \rangle, \textsc{Scd}, \textsc{Obs} \rangle \Vdash \langle \mathcal{N}, \mathcal{O}, \textsc{Qlaw}(\textsc{Scd}) \rangle$$

or, equivalently,

$$\langle \mathcal{Q}, \mathcal{O}, \langle \Psi, \textsc{True}, \textsc{Elaw} \rangle, \textsc{Scd}, \textsc{Obs} \rangle \Vdash \langle \mathcal{N}, \mathcal{O}, \textsc{Qlaw}(\textsc{Scd}) \rangle$$

then the four scenario descriptions

- $\langle \mathcal{K}, \mathcal{O}, \langle \Psi, \textsc{Elaw} \rangle, \textsc{Scd}, \textsc{Obs} \rangle$
- $\langle \mathcal{K}, \mathcal{O}, \langle \Psi, \textsc{Law} \rangle, \textsc{Scd}, \textsc{Obs} \rangle$
- $\langle \mathcal{Q}, \mathcal{O}, \langle \Psi, \textsc{True}, \textsc{Elaw} \rangle, \textsc{Scd}, \textsc{Obs} \rangle$
- $\langle \mathcal{Q}, \mathcal{O}, \langle \Psi, \textsc{Qlaw}, \textsc{Elaw} \rangle, \textsc{Scd}, \textsc{Obs} \rangle$

all represent the same set of games.

Corollary 1 *Let every time-point in* $\textsc{Goal}$ *be equal to or later than any time-point in* $\textsc{Scd}$ *and* $\textsc{Obs}$*, let* $\Psi = \textbf{IA}$ *and let* $\textsc{Law}$ *be equivalent to* $\textsc{Elaw}$ *and* $\textsc{Qlaw}$*.*

$$\langle \mathcal{Q}, \mathcal{O}, \langle \Psi, \textsc{Qlaw}, \textsc{Elaw} \rangle, \textsc{Scd}, \textsc{Obs} \rangle \Vdash \langle \mathcal{N}, \mathcal{O}, \textsc{Goal} \rangle$$

iff

$$\langle \mathcal{K}, \mathcal{O}, \langle \Psi, \textsc{Elaw} \rangle, \textsc{Scd}, \textsc{Obs} \rangle \Vdash \langle \mathcal{N}, \mathcal{O}, \textsc{Qlaw}(\textsc{Scd}) \cup \textsc{Goal} \rangle.$$

Let us consider corollary 1. In the scenario description

$$\langle \mathcal{K}, \mathcal{O}, \langle \Psi, \textsc{Elaw} \rangle, \textsc{Scd}, \textsc{Obs} \rangle$$

all actions in $\textsc{Scd}$ are executable. The set $\textsc{Scd}$ corresponds to the plan. The qualifications, or executability conditions, have been moved to the goal description

$$\langle \mathcal{N}, \mathcal{O}, \textsc{Qlaw}(\textsc{Scd}) \cup \textsc{Goal} \rangle$$

and, consequently, have no effect on the plan. Apparently, here we have a possibility during the planning process to reason about a *partial* plan in $\mathcal{Q}$

$$\langle \mathcal{Q}, \mathcal{O}, \langle \Psi, \textsc{Qlaw}, \textsc{Elaw} \rangle, \textsc{Scd}, \textsc{Obs} \rangle$$

as if it were executable, by transforming it to a $\mathcal{K}$ scenario as above. A similar technique can be applied when planning in $\mathcal{K}$. The action laws in $\textsc{Law}$ are split into two parts, forming an equivalent pair $\textsc{Elaw}$ and $\textsc{Qlaw}$.

2.7 An Example

Consider the following plan synthesis problem in $\mathcal{Q}$-**IA**. The world

$$\langle \mathcal{Q}, \mathcal{O}, \langle \mathbf{IA}, \textsc{Qlaw}, \textsc{Elaw} \rangle, \emptyset, \textsc{Obs} \rangle$$

has the following contents:

$$\begin{array}{ll}
\mathsf{qlaw1} & [t,t'] \, EnterRestaurant \rightsquigarrow [t] \, inStreet \\
\mathsf{elaw1} & [t,t'] \, EnterRestaurant \rightsquigarrow \\
 & \quad [t,t'] \, inStreet{:=}F \wedge [t,t'] \, inRest{:=}T \\
\mathsf{qlaw2} & [t,t'] \, Dine \rightsquigarrow [t] \, inRest \\
\mathsf{elaw2} & [t,t'] \, Dine \rightsquigarrow [t,t'] \, hungry{:=}F \\
\mathsf{obs1} & [0] \, hungry \\
\mathsf{obs2} & [0] \, inStreet \\
\mathsf{obs3} & [0] \, \neg inRest
\end{array} \tag{29}$$

The goal $\langle \mathcal{N}, \mathcal{O}, \textsc{Goal} \rangle$ has the following contents:

$$\mathsf{goal1} \quad [10] \, \neg hungry \tag{30}$$

We start with a schedule $\textsc{Scd} = \emptyset$ representing the plan. By a transformation to $\mathcal{K}$, we can conclude that

$$\langle \mathcal{K}, \mathcal{O}, \langle \mathbf{IA}, \textsc{Elaw} \rangle, \textsc{Scd}, \textsc{Obs} \rangle \mid\!\sim \langle \mathcal{N}, \mathcal{O}, \{ [4] \, hungry \} \rangle$$

In order to achieve $\{ [4] \, \neg hungry \}$, an action is added to the plan. The new schedule is called $\textsc{Scd}'$.

$$\mathsf{scd1} \quad [7,9] \, Dine \tag{31}$$

Now we can observe that

$$\langle \mathcal{K}, \mathcal{O}, \langle \mathbf{IA}, \textsc{Elaw} \rangle, \textsc{Scd}', \textsc{Obs} \rangle \mid\!\sim \langle \mathcal{N}, \mathcal{O}, \{ [10] \, \neg hungry \} \rangle$$

However, $\textsc{Qlaw}(\textsc{Scd}) = \{ [7] \, inRest \}$ and

$$\langle \mathcal{K}, \mathcal{O}, \langle \mathbf{IA}, \textsc{Elaw} \rangle, \textsc{Scd}', \textsc{Obs} \rangle \mid\!\sim \langle \mathcal{N}, \mathcal{O}, \{ [7] \, \neg inRest \} \rangle$$

In order to achieve $\{[7]\,inRest\}$, another action is added to the plan.

$$\begin{array}{ll} \mathsf{scd1} & [2,5]\,EnterRestaurant \\ \mathsf{scd2} & [7,9]\,Dine \end{array} \tag{32}$$

Now, $\mathrm{Q}_{\mathrm{LAW}}(\mathrm{S}_{\mathrm{CD}}'') = \{[2]\,inStreet,\ [7]\,inRest\}$. Furthermore,

$$\begin{array}{l} \langle \mathcal{K}, \mathcal{O}, \langle \mathbf{IA}, \mathrm{E}_{\mathrm{LAW}} \rangle, \mathrm{S}_{\mathrm{CD}}'', \mathrm{O}_{\mathrm{BS}} \rangle \mid\!\sim \\ \qquad \langle \mathcal{N}, \mathcal{O}, \{[2]\,inStreet,\ [7]\,inRest,\ [10]\neg hungry\} \rangle \end{array}$$

Thus, the plan achieves both the goal and the qualifications of all actions. By theorem 2, the plan is executable and achieves the goal. The solution is

$$\langle \mathcal{K}, \mathcal{O}, \langle \mathbf{IA}, \mathrm{E}_{\mathrm{LAW}} \rangle, \{[2,5]\,EnterRestaurant, [7,9]\,Dine\}, \emptyset \rangle.$$

3 STRIPS: A Total-Order Planner

3.1 Precondition-Achievement Operator Planning

In operator planning, the world is modelled in terms of situations and operators. What is true in a situation is usually expressed by a set of sentences in some formal language. An operator is basically a function from situations to situations. The traditional way of modelling an operator is by preconditions that have to be true in order for the operator to be applicable, and effects that state how the operator modifies the state of the world.

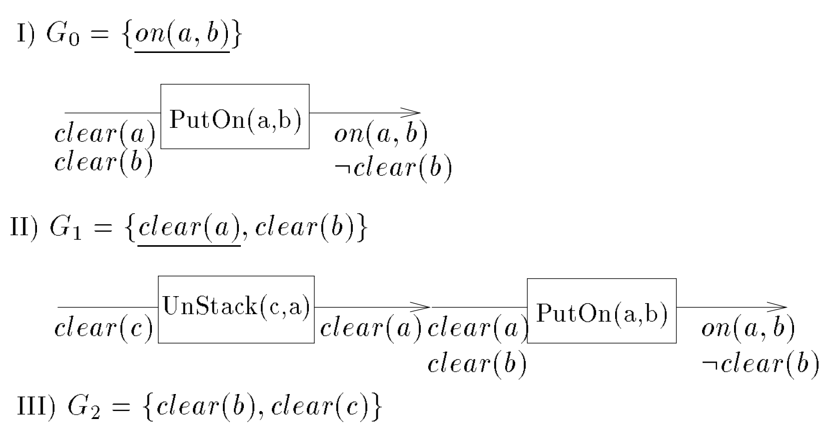

Figure 5: Precondition-achievement planning. G_n is the set of goals at the nth step in the planning process; the next selected goal is underlined.

One approach to operator planning is by precondition-achievement. The principle is straight-forward. If there is a goal g which should be achieved and an operator α that has g among its effects, then α can be added to the plan and the preconditions $p_1, \ldots, p_n$ of α will be treated as new goals in the planning process (fig. 5).

In this section, we analyse and reconstruct the total-order planner STRIPS [FN71]. In the next section, we will do the same thing for a partial-order planner, TWEAK[Cha87]. Another partial-order planner, which recently has earned much attention, is McAllester's and Rosenblitt's *systematic non-linear planner* (SNLP) [MR91]. As SNLP has the same operator representation as STRIPS, the results in these sections applies for SNLP as well.

3.2 STRIPS — a Total-Order Planner

STRIPS is a total-order planner. The search in the plan space is done from the end of the plan, in a regressive manner. However, a step is never added to the plan until all its preconditions have been made true. When a step that asserts a specific goal or subgoal has been selected, its preconditions are treated as new goals to fulfill, and first when this has been done is the step added to the plan. Steps are only added to the end of the plan, and consequently the plan is built up in a progressive manner. An advantage of this mode of procedure is that the resulting partial plans are always executable.

We will now make an attempt to characterize the semantics underlying the STRIPS assumption, and from there move on to a STRIPS in Fluent Logic. As the original article by Fikes and Nilsson [FN71] only gives a syntactic characterization of operators and states, we will have to start from this and try to translate it to a state-based semantics. The formal description of STRIPS in the following subsections is inspired by Lifschitz [Lif87].

3.2.1 World Models and States

Fikes and Nilsson use *world models* to describe states. A world model is represented by a set of well-formed closed FOPC formulae. There appears to be no restriction that this set of formulae should give a complete description of the world. Consequently, world models will be interpreted as *partial states* (or sets of states). An alternative approach is to use a *closed world assumption* [GN87].

3.2.2 STRIPS Operators and Plans

A STRIPS *operator* α is a transformation on world models. An operator description for α is a triple $(Pre_\alpha, Del_\alpha, Add_\alpha)$. Pre_α is the description of the *preconditions* which have to be true for the operator to be applicable, Del_α is the *delete list*, i.e. a set of formulae which are no longer true after the application of α, and Add_α is the *add list*, i.e. formulae which become true. The set of available operators is denoted O.

For convenience, sets of operators representing the same type of action but with different objects involved are abstracted to operator schemas. We illustrate with an example of an operator schema:

$$\begin{aligned} Operator: &\quad Pickup(x) \\ Pre: &\quad onTable(x) \wedge handEmpty(x) \wedge clear(x) \\ Del: &\quad onTable(x), handEmpty(x), clear(x) \\ Add: &\quad holding(x) \end{aligned}$$

The description of the operator $Pickup(Block1)$ is obtained by substituting $Block1$ for x above.

The applicability of an operator α to a world model M can be defined as follows:

$$Applicable(\alpha, M) \equiv M \vdash Pre_\alpha \tag{33}$$

and the result of applying α to a world model M can be defined as follows:

$$Result(\alpha, M) = (M \setminus Del_\alpha) \cup Add_\alpha \tag{34}$$

The latter formula is the STRIPS solution to the frame problem: everything in M which is not in Del_α persists. Observe that if $\delta \in Del_\alpha$, this does not imply that $\neg\delta \in Result(\alpha, M)$; if $\neg\delta \notin Add_\alpha$, then δ might be undefined in $Result(\alpha, M)$. The result of the application of an unapplicable operator is undefined.

A plan $\overline{\alpha}$ is a sequence of operators $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ where each operator occurrence is called a *step*. An operator may occur at more than one step in a plan. The result of a plan is the result of the successive applications of the steps in the plan.

$$\begin{aligned} &Result((), M) = M \\ &Result((\alpha_1, \alpha_2, \ldots, \alpha_n), M) = \\ &\qquad\qquad Result((\alpha_2, \ldots, \alpha_n), Result(\alpha_1, M)) \end{aligned} \tag{35}$$

Applicability of plans is defined in a similar manner.

$$\begin{aligned} &Applicable((), M) \equiv True \\ &Applicable((\alpha_1, \alpha_2, \ldots, \alpha_n), M) \equiv \\ &\qquad Applicable(\alpha_1, M) \wedge Applicable((\alpha_2, \ldots, \alpha_n), Result(\alpha_1, M)) \end{aligned} \tag{36}$$

The successive application of the steps in the plan defines a sequence of world models. A *situation* is an entity associated with a position in the plan and the corresponding world model. The situation following a step is associated with the world model resulting from the application of the plan up to and including that particular step. The *initial situation* is associated with the initial world model, and the *goal situation* is the situation following the last step in the plan.

3.2.3 STRIPS System and Planning Problem

A STRIPS system Σ is a tuple

$$\langle M_0, \{\, (Pre_\alpha, Del_\alpha, Add_\alpha) \mid \alpha \in O \,\}\rangle$$

where M_0 is the world model describing the initial state of the world, and O is the set of available operators.
A STRIPS planning problem is a tuple

$$\langle \Sigma, G \rangle$$

where Σ is a STRIPS system as above and G is a formula constituting the *goal description*. Fikes and Nilsson specify no restrictions on G.
A plan $\overline{\alpha}$ is a solution iff

1. For all α_i in $\overline{\alpha}$, $\alpha_i \in O$.
2. $Applicable(\overline{\alpha}, M_0)$.
3. $Result(\overline{\alpha}, M_0) \vdash G$.

3.2.4 Soundness and Essential Formulae

Operators are defined as *syntactic* transformations of world models. This is problematic; it is easy to construct STRIPS systems that are unsound. Lifschitz [Lif87] suggests selecting a set E of formulae as *essential.* Only essential formulae are allowed to change by means of Del and Add; non-essential formulae occurring in M_0 (or in the add- or deletelist of any operator) have to be true in all world models. The choice of E is an important aspect of the design of a STRIPS system. Typically, essential formulae are selected as atomic formulae or literals [3] representing the dynamical properties of the world, e.g. $open(door1)$. Typical non-essential formulae are $connects(Room1, Room2)$, $\forall x.[connects(x, y) \Rightarrow connects(y, x)]$ and $open(Door1) \vee in(Box1, Room4)$. The two first formulae state statical conditions, whereas the last one is simply not a literal.

Lifschitz models the semantics of the world in terms of states, where s_0 is the initial state, and actions, which are partial functions from states to states. An action f is applicable in a state s if $f(s)$ is defined, and in that case $f(s)$ is the result of the action. An action f_α is associated with each operator α. A plan $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ corresponds to a function composition $f_{\alpha_n} \circ \ldots \circ f_{\alpha_1}$. Finally, a world model M is satisfied in a state s if every element of M is satisfied in s.

Lifschitz definition of soundness is as follows.[4]

Definition 5 [Lif87] *An operator description* (Pre, Del, Add) *is* **sound** *relative to an action* f *if, for every state* s *such that* P *is satisfied in* s*,*

1. f *is applicable in state* s*,*
2. *every essential sentence which is satisfied in* s *and does not belong to* D *is satisfied in* $f(s)$*,*
3. A *is satisfied in* $f(s)$*,*
4. *every sentence in* A *which is not essential is satisfied in all states of the world.*

(A STRIPS *system)* Σ *is* **sound** *if*

1. M_0 *is satisfied in the initial state* s_0*,*
2. *every sentence in* M_0 *which is not essential is satisfied in all states of the world,*
3. *every operator description* $(Pre_\alpha, Del_\alpha, Add_\alpha)$ *is sound relative to* f_α*.*

Theorem 3 [Lif87] *If* Σ *is sound, and a plan* $\overline{\alpha}$ *is accepted by* Σ*, then the action* $f_{\overline{\alpha}}$ *is applicable in the initial state* s_0 *and the world model* $Result(\overline{\alpha}, M_0)$ *is satisfied in the state* $f_{\overline{\alpha}}(s_0)$*.*

[3] A literal is either an atomic formulae or a negated atomic formulae

[4] The following definitions and theorem have been slightly modified to suit our notation.

3.2.5 The STRIPS Algorithm

The STRIPS algorithm is straight-forward. The recursive version presented here is based upon a version by Nilsson [Nil80], but has been modified to suit our notation.

The symbol Π denotes a (partial) plan, S denotes a world model, and finally G is a set of goal sentences. The function *concat* performs plan concatenation.

function $STRIPS(S, G)$
1 $\Pi \leftarrow ()$
2 until $S \vdash G$ do
3 $\quad g \leftarrow$ nondeterministically chosen element γ of G such that not $S \vdash \gamma$
4 $\quad o \leftarrow$ nondeterministically chosen operator $\alpha \in O$ such that $Add_\alpha \vdash g$
5 $\quad \langle S, \Pi' \rangle \leftarrow STRIPS(S, Pre_o)$
6 $\quad \Pi \leftarrow concat(\Pi, \Pi')$
7 return $\langle Result(o, S), concat(\Pi, (o)) \rangle$

Observe that the STRIPS presented so far is totally ground. A ground operator is always selected at step 4; this is a strategy with a high level of early commitment. However, it is easy to extend the algorithm to incorporate *partially instantiated operators* at step 4. Free variables represent the unspecified arguments of the operator. When the preconditions are passed on as new goals in the recursive calls, the algorithm will find substitutions for the variables, and a fully instantiated operator is eventually inserted at step 7. The plan Π should always be ground.

3.3 Epistemology and Ontology of STRIPS

The result of an unapplicable operator is undefined in STRIPS. Furthermore, all observations are from the initial state and all plans are ground. Consequently, STRIPS belongs to the $\mathcal{Q}_{\mathsf{pg}}$ epistemological class.

The essential formulae can be selected in many ways. Therefore, it is possible to construct systems with quite complicated ontological properties. We will therefore classify STRIPS systems according to the choice of essential formulae, but also by what formulae are allowed in M_0 and how D and A are related. No restrictions will be put on P or goals.

We shall start with the following restricted class of STRIPS systems.

Definition 6 *A* **STRIPS-I system** *is a* STRIPS *system with the following restrictions.*

1. *All formulae and operators are ground and function-free.*
2. *The essential formulae are atomic formulae and their negations.*
3. *For all $\alpha \in O$, $\delta \in Del_\alpha$ if and only if $\neg\delta \in Add_\alpha$, and $\neg\delta \in Del_\alpha$ if and only if $\delta \in Add_\alpha$.*

4. *Non-essential formulae in M_0 must not contain any essential subformulae.*

These restrictions give us systems where essential formulae change truth values in an unambiguous way, and only if they occur in Del_α or Add_α. Expressing this in terms of underlying semantics, each action has one unique trajectory. There are no alternative results or ramifications.

Removing restriction 3 results in ambiguous change of postcondition truth values; this corresponds to random action results and implies speciality **A**. For instance, the operator $\alpha = (\{\}, \{P(a)\}, \{\})$ leaves the truth value of $P(a)$ unspecified in the resulting situation, and this corresponds to two sets of states; one set where $P(a)$ is true and one set where $P(a)$ is false.

Removing restriction 2 gives a more unpredictable result. Some examples:

- If $Add_\alpha = \{P(x) \vee Q(x), \ldots\}$, we might have random results (speciality **A**).
- If $Add_\alpha = \{P(x) \Rightarrow Q(x), \ldots\}$, we might have local ramification (speciality **U**).
- If $Add_\alpha = \{\forall y[R(x,y) \Rightarrow Q(y)], \ldots\}$, we might have structural ramification (speciality **D**).

In a similar manner, removing restriction 4 might lead to local or structural ramification (read M_0 for Add_α above).

Observe that this doesn't imply that the STRIPS assumption is expressive enough to fully cover these specialities. For instance, STRIPS is incapable of expressing context-dependent results of actions, which is a typical characteristic of speciality **A**. It does however imply that some STRIPS systems might be within these specialities.

3.4 A Logicist Approach to STRIPS-I

The STRIPS representation belongs to the epistemological speciality $\mathcal{Q}_{\mathsf{pg}}$. We shall use an intermediate transformation to $\mathcal{K}_{\mathsf{pg}}$ as described in the previous section. $\mathcal{K}_{\mathsf{pg}} - \mathbf{Isb}$ is within the range of applicability of PMON [San94].

The time domain is the non-negative integers $\mathbb{Z}$. Furthermore, a set of *fluents* are selected as *essential*.
The world and the plan is a FL chronicle

$$\langle \mathcal{K}, \mathcal{O}, \textsc{Elaw}, \textsc{Scd}, \textsc{Obs} \rangle. \tag{37}$$

ELAW represents the action effect laws in the form

$$\begin{aligned} &\mathsf{elaw} \quad [t,t']\, A(\overline{x}) \leadsto \\ &\qquad [t,t']\, p_1(\overline{x}_1) := B_1 \wedge \ldots \wedge [t,t']\, p_n(\overline{x}_n) := B_n \end{aligned} \tag{38}$$

where $p_i(\overline{x}_i)$ are *essential* fluents and $B_i \in \{T,\ F\}$.

$\textsc{Scd}$ is the plan; a set of sequential actions of the form:

$$\textsf{scd} \quad [\boldsymbol{t}, \boldsymbol{t}'] \, A_i(\overline{\boldsymbol{c}}_i) \tag{39}$$

where $\boldsymbol{t}, \boldsymbol{t}' \in \mathbb{Z}$ and $\boldsymbol{t}' = \boldsymbol{t} + 1$.
$\textsc{Obs}$ is the initial state and state constraints; a set of formulae of the form ($[\neg]$ denotes possible negation):

$$\textsf{obs} \quad [0] \, [\neg] p(\overline{\boldsymbol{c}}) \tag{40}$$

or the form:

$$\textsf{obs} \quad [\,] \, \beta \tag{41}$$

where $[\,]\beta$ contains no essential fluents.
The goal and preconditions is a scenario description

$$\langle \mathcal{N}, \textsc{Goal} \cup \textsc{Qlaw}(\textsc{Scd}) \rangle \tag{42}$$

$\textsc{Goal}$ is the goals:

$$\textsf{goal} \quad [S_0] \, \alpha(\overline{c}) \tag{43}$$

The goals may contain both essential and non-essential fluents. The unfixed temporal constant S_0 implies that the time for the goal is unspecified.
$\textsc{Qlaw}$ represents the qualification laws in the form:

$$\textsf{qlaw} \quad [t, t'] \, A(\overline{x}) \leadsto [t] \, \beta(\overline{x}) \tag{44}$$

where $[t] \, \beta(\overline{x})$ is the precondition description.

3.4.1 A Truth Criterion for STRIPS-I

The concept of truth criteria was first introduced by Chapman [Cha87] for his partial-order planner TWEAK, which is presented in the next section. The main purpose of a truth criterion is to support efficient computation of whether a predicate is true or false in a particular situation in the plan. Furthermore, if it is known how to decide the truth value of a predicate in a situation, this knowledge can be used to decide how to make a predicate true in a situation. The truth criterion of TWEAK was reformulated to a *goal achievement procedure*.

We will now formulate a truth criterion for STRIPS. It could be argued that the truth criterion for TWEAK is necessary to cope with the complexity inherit in partial-order plans, but that it is unnecessary for a total order planner like STRIPS. We do not state that the truth criterion for STRIPS presented here is indispensable, but we do think it is of principal interest.

In plain language, the criterion says that $p(\overline{\boldsymbol{c}})$ is true at $\boldsymbol{t}$ iff there is a time-point $\boldsymbol{t}'$ such that 1) $p(\overline{\boldsymbol{c}})$ is made true at $\boldsymbol{t}'$, either by an action or in the initial state ($\boldsymbol{t}' = 0$), and 2) there is no intervening time-point where $p(\overline{\boldsymbol{c}})$ is affected by some action ($\neg p^*(\overline{\boldsymbol{c}})$ means that $p(\overline{\boldsymbol{c}})$ is not occluded, and therefore persists).

Theorem 4 (Truth Criterion for Ground Linear Plans) *Let P be a chronicle $\langle \mathcal{K}, \mathcal{O}, \textsc{Elaw}, \textsc{Scd}, \textsc{Obs} \rangle$ as above, and let $[\boldsymbol{t}]\, p(\overline{\boldsymbol{c}})$ be an essential formula.*

$$P \mid\!\sim \ulcorner [\boldsymbol{t}]\, p(\overline{\boldsymbol{c}}) \urcorner \text{ iff } \exists \boldsymbol{t}'[\ [\ \boldsymbol{t}' \leq \boldsymbol{t} \wedge P \mid\!\sim \ulcorner [\boldsymbol{t}'-1, \boldsymbol{t}']\, p(\overline{\boldsymbol{c}}) {:=} T \urcorner \vee \\ \boldsymbol{t}' = 0 \wedge P \mid\!\sim \ulcorner [\boldsymbol{t}']\, p(\overline{\boldsymbol{c}}) \urcorner\ ] \wedge \\ \forall \boldsymbol{t}''[\ \boldsymbol{t}' < \boldsymbol{t}'' \leq \boldsymbol{t} \Rightarrow P \mid\!\sim \ulcorner [\boldsymbol{t}']\, \neg p^*(\overline{\boldsymbol{c}}) \urcorner\ ]] \tag{45}$$

For negative literals, substitute $\neg p(\overline{\boldsymbol{c}})$ for $p(\overline{\boldsymbol{c}})$ and F for T above.

The truth criterion reflects how STRIPS achieves a single, atomic goal. The reassignment statement corresponds to a preceding action asserting the literal goal. The negated occlusion do not have any explicit correspondence in the algorithm, but if it is violated, the algorithm might have to reachieve the goal in question.

3.4.2 The STRIPS Algorithm — FL version

The overall structure of the algorithm can be maintained. However, some of the operations in the algorithm need to be reformulated.

Plan concatenation. Each step of the FL plan is associated with a time-point. Keeping track of the latest time-point can be done in the same way as the original STRIPS keeps track of the last situation S. Having assigned proper time-points to each step, the concatenation is simply a union of the two plans.

The current world model S. In original STRIPS, a full description of the current S is always kept. With FL, it is easy to reason backwards in time to decide whether a statement is true at a time-point or not, or to use the truth criterion. Naturally, it is possible to explicitly represent all facts holding in the current situation.

An algorithm for FL-STRIPS can now be formulated. Let $\gamma(\boldsymbol{t})$ denote the temporal formula where all time-points in γ are set to $\boldsymbol{t}$.

function $STRIPS(\boldsymbol{t}, \Gamma, G)$
1 until $\Gamma \mid\!\sim G(\boldsymbol{t})$ do
2 $\quad g \leftarrow$ nondeterministically chosen element $[S]\,\gamma$ of G such that not $\Gamma \vdash [\boldsymbol{t}]\,\gamma$
3 $\quad o \leftarrow$ nondeterministically chosen action $[S', S'+1]\, A$ such that $\textsc{Elaw}([S', S'+1]\, A) \vdash g(S'+1)$
4 $\quad \langle \boldsymbol{t}, \Gamma \rangle \leftarrow STRIPS(\boldsymbol{t}, \textsc{Qlaw}([S', S'+1]\, A))$
5 return $\langle \boldsymbol{t}+1, \Gamma \cup \{[\boldsymbol{t}, \boldsymbol{t}+1]\, A\} \rangle$

3.4.3 An Example

We will illustrate the functionality of FL-STRIPS with a blocks-world example.

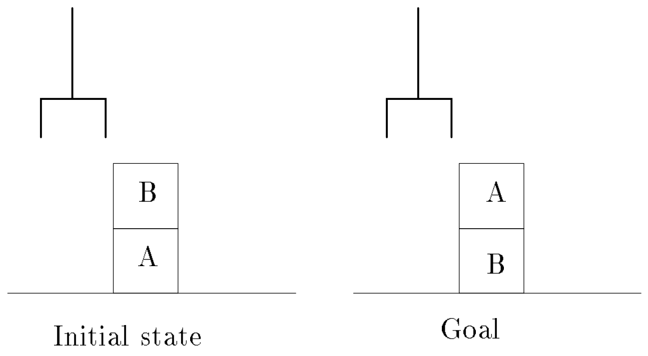

Figure 6: A blocks-world planning problem.

Assume we have the following world Γ_W (fig. 3.4.3).

$$
\begin{array}{ll}
\textsf{qlaw1} & [t,t']\,Pickup(x) \rightsquigarrow \\
 & \quad [t]\,clear(x) \wedge [t]\,onTable(x) \wedge [t]\,handEmpty \\
\textsf{elaw1} & [t,t']\,Pickup(x) \rightsquigarrow \\
 & \quad [t,t']\,clear(x){:=}F \wedge [t,t']\,onTable(x){:=}F \\
 & \quad \wedge\, [t,t']\,handEmpty{:=}F \wedge [t']\,holding(x){:=}T \\
\textsf{qlaw2} & [t,t']\,Putdown(x) \rightsquigarrow [t]\,holding(x) \\
\textsf{elaw2} & [t,t']\,Putdown(x) \rightsquigarrow \\
 & \quad [t,t']\,clear(x){:=}T \wedge [t,t']\,onTable(x){:=}T \\
 & \quad \wedge\, [t,t']\,handEmpty{:=}T \wedge [t,t']\,holding(x){:=}F \\
\textsf{qlaw3} & [t,t']\,Unstack(x,y) \rightsquigarrow \\
 & \quad [t]\,clear(x) \wedge [t]\,on(x,y) \wedge [t]\,handEmpty \\
\textsf{elaw3} & [t,t']\,Unstack(x,y) \rightsquigarrow \\
 & \quad [t,t']\,clear(x){:=}F \wedge [t,t']\,clear(y){:=}T \\
 & \quad \wedge\, [t,t']\,on(x,y){:=}F \wedge [t,t']\,handEmpty{:=}F \\
 & \quad \wedge\, [t,t']\,holding(x){:=}T \\
\textsf{qlaw4} & [t,t']\,Stack(x,y) \rightsquigarrow [t]\,clear(y) \wedge [t]\,holding(x) \\
\textsf{elaw4} & [t,t']\,Stack(x,y) \rightsquigarrow \\
 & \quad [t,t']\,clear(x){:=}T \wedge [t,t']\,clear(y){:=}F \\
 & \quad \wedge\, [t,t']\,on(x,y){:=}T \wedge [t,t']\,handEmpty{:=}T \\
 & \quad \wedge\, [t,t']\,holding(x){:=}F \\
\textsf{obs1} & [0]\,onTable(A) \wedge [0]\,on(B,A) \wedge [0]\,\neg clear(A) \\
 & \wedge\, [0]\,clear(B) \wedge [0]\,handEmpty
\end{array}
\tag{46}
$$

The goal Γ_G is as follows:

$$
\textsf{goal1} \quad [S_0]\,onTable(B) \wedge [S_0]\,on(A,B) \tag{47}
$$

The planning process progresses as follows. For convenience, we will assume that the planner always makes the right choice at a nondeterministic choice-point.

1. Initially, $\boldsymbol{t} = 0$ and the plan Γ is empty.

2. STRIPS tries to achieve $[S_0]\,onTable(B) \wedge [S_0]\,on(A, B)$.
   The component $[S_0]\,onTable(B)$ is selected.
   The action $[S_1, S_1 + 1]\,Putdown(B)$ is selected:
   QLAW($[S_1, S_1 + 1]\,Putdown(B)) \vdash [S_1 + 1]\,onTable(B)$.
   The planner is recursively called with the new goal $[S_1]\,holding(B)$.

3. Goal: $[S_1]\,holding(B)$.
   Select: $[S_1]\,holding(B)$.
   Action: $[S_2, S_2 + 1]\,Unstack(B, A)$.
   New goal: $[S_2]\,clear(B) \wedge [S_2]\,on(B, A) \wedge [S_2]\,handEmpty$.

4. Goal: $[S_2]\,clear(B) \wedge [S_2]\,on(B, A) \wedge [S_2]\,handEmpty$.
   Clearly, $\Gamma_W \cup \Gamma \mathrel{|\!\!\sim} [0]\,clear(B) \wedge [0]\,on(B, A) \wedge [0]\,handEmpty$.

5. Proceeding from 3.
   Action
   scd1 $[0, 1]\,Unstack(B, A)$
   can now be added to the plan Γ', and $\boldsymbol{t} = 1$.
   Remaining goal: $[S_1]\,holding(B)$.
   Clearly, $\Gamma_W \cup \Gamma' \mathrel{|\!\!\sim} [1]\,holding(B)$.

6. Proceeding from 2.
   Action
   scd2 $[1, 2]\,Putdown(B)$
   can now be added to the plan Γ', and $\boldsymbol{t} = 2$.
   Clearly, $\Gamma_W \cup \Gamma' \mathrel{|\!\!\sim} [2]\,onTable(B)$.
   The first part of goal1 has now been achieved.
   Remaining goal: $[S_0]\,on(A, B)$.
   Select: $[S_0]\,on(A, B)$.
   Action: $[S_3, S_3 + 1]\,Stack(A, B)$.
   New goal: $[S_3]\,clear(B) \wedge [S_3]\,holding(A)$.

7. Goal: $[S_3]\,clear(B) \wedge [S_3]\,holding(A)$.
   Select: $[S_3]\,holding(A)$.
   Action: $[S_4, S_4 + 1]\,Pickup(A)$.
   New goals: $[S_4]\,clear(A) \wedge [S_4]\,onTable(A) \wedge [S_4]\,handEmpty$.

8. Goal: $[S_4]\,clear(A) \wedge [S_4]\,onTable(A) \wedge [S_4]\,handEmpty$.
   Clearly, $\Gamma_W \cup \Gamma' \mathrel{|\!\!\sim} [2]\,clear(A) \wedge [2]\,onTable(A) \wedge [2]\,handEmpty$.

9. Proceeding from 7.
   Action
   scd3 $[2, 3]\,Pickup(A)$
   can now be added to the plan Γ', and $\boldsymbol{t} = 3$.
   Remaining goal: $[S_3]\,clear(B) \wedge [S_3]\,holding(A)$.
   Clearly, $\Gamma_W \cup \Gamma' \mathrel{|\!\!\sim} [3]\,clear(B) \wedge [3]\,holding(A)$.

10. Proceeding from 6.
    Action
    $\mathsf{scd4}$ $[3,4]\,Stack(A,B)$
    can now be added to the plan Γ', and $\boldsymbol{t} = 4$
    Clearly, $\Gamma_W \cup \Gamma' \mathrel{|\!\!\sim} [4]\,onTable(B) \wedge [4]\,on(A,B)$.
    Both parts of **goal1** have now been achieved (let $S_0 = 4$).

The plan Γ' is

$$\begin{array}{ll} \mathsf{scd1} & [0,1]\,Unstack(B,A) \\ \mathsf{scd2} & [1,2]\,Putdown(B) \\ \mathsf{scd3} & [2,3]\,Pickup(A) \\ \mathsf{scd4} & [3,4]\,Stack(A,B). \end{array} \tag{48}$$

It is easily verified that the plan is executable and achieves the goal.

4 TWEAK — a Partial-Order Planner

The TWEAK planner [Cha87] is based on the *least commitment* principle, stating that during the planning process, decisions about for instance the order between two steps should not be made until they have to be made. TWEAK has two new major features compared to STRIPS.

1. The plans of TWEAK are *partially ordered.* The order among the steps, or actions, in a TWEAK plan need not be totally specified. Instead of immediate precedence, the order among steps is expressed in terms of "before", like in $s_1 < s_2$ where s_1 and s_2 are steps.

2. Steps in a plan can be *partially instantiated.* Variables can be used to represent objects in a step. A *codesignation relation* constrains the possible values of a variable. For instance, $x \approx a$ denotes that the variable x stands for the object a, and $x \not\approx y$ denotes that each of the variables x and y represents a different object.

The semantics of a TWEAK plan is defined in terms of its *completions*, that is the set of complete plans one can get by imposing on the plan a total order and a value for each object variable. Each such completion has a well-defined semantic evaluation. The plan is then interpreted in terms of its set of completions. A statement γ is *necessarily true* if it is true in *all* completions of the plan (written $\Box\gamma$), and it is *possibly true* if it is true in *some* completion (written $\Diamond\gamma$).

4.0.4 World models

On the completion level, TWEAK has a representation similar to STRIPS, but a more restricted one. World models consist of sets of literals (Chapman uses the term *propositions*). A world model is a description of a partial state.

The object domain is assumed to be infinite. Naturally, there are only a finite number of objects mentioned in a planning problem, but an infinite object domain is needed for the completeness of the TWEAK algorithm.

4.0.5 TWEAK Operators and Plans

Operators are tuples $\alpha = (Pre, Post)$ of sets of precondition and postcondition literals. The postconditions are said to be *asserted* in the world model resulting from applying the operator, and their negations are said to be *denied* there. Variables are allowed in operator instances.

Let θ be a substitution where the corresponding objects are substituted for the variables in the current completion. Applicability of an operator at a situation is defined as follows:

$$Applicable(\alpha\theta, M) \equiv Pre\theta \subseteq M \qquad (49)$$

Let the denial operator $\sim$ be defined as $\sim p \equiv \neg p$ and $\sim \neg p \equiv p$ for atomic formulae p, and let θ be as before. The result of applying an operator α to a world model M in a completion is defined as follows:

$$Result(\alpha\theta, M) = (M \setminus \{\sim \delta | \delta \in Post\theta\}) \cup Post\theta \tag{50}$$

A TWEAK plan P is a tuple

$$\langle S, C, T \rangle \tag{51}$$

where

- S is a set of *steps*, which are occurrences of operators.
- C is a set of codesignation constraints on the variables appearing in S and the objects in the object domain.
- T is a set of ordering constraints on the steps in S.

4.0.6 Completions and Modalities

A plan P has a set $Comp(P)$ of completions $\langle S, C', T' \rangle$ where

- C' is a minimal set of codesignation constraints such that $C \subseteq C'$, C' is consistent and C' makes each variable in each step codesignate with some object. We shall call the corresponding substitution $\theta_{C'}$.
- T' is a minimal set of ordering constraints such that $T \subseteq T'$, T' is consistent and for each pair s_i, $s_j \in S$, $\{s_i < s_j\} \in T'$ or $\{s_j < s_i\} \in T'$.

A completion corresponds to a ground linear plan $\overline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, where each α_i is a ground operator. The definitions of *Result* and *Applicable* are extended to handle ground linear plans, just like in STRIPS.

A *situation* is an entity associated with a position in the plan and the corresponding world model. The concepts of assertion and denial are applicable to situations and steps.

Necessary and possible truth can now be given the following definitions.

$$\Box\gamma \text{ in } P \equiv \forall P' \in Comp(P).[\gamma \text{ in } P'] \tag{52}$$

$$\Diamond\gamma \text{ in } P \equiv \exists P' \in Comp(P).[\gamma \text{ in } P'] \tag{53}$$

Chapman states no restrictions on the nature of γ. We will restrict γ to statements about individual steps, situations and constraints.[5]

[5] With no restrictions on γ, strange things may happen. For instance, all completions of a plan are totally ordered and instantiated, and therefore the plan itself is necessarily totally ordered and instantiated [KN94].

4.0.7 TWEAK System and Planning Problem

A TWEAK system Σ is a tuple

$$\langle M_0, O \rangle \tag{54}$$

where M_0 is the initial world model, and O is a set of operators.

A TWEAK planning problem is a tuple

$$\langle \Sigma, G \rangle \tag{55}$$

where Σ is a TWEAK system and G is a set of literals representing the goal. A set of literals is equivalent to a conjunction of literals, and therefore TWEAK is called a conjunctive planner.

A plan P is a solution to a problem iff for each completion $\overline{\alpha} \in Comp(P)$ the following holds.

1. For all α_i in $\overline{\alpha}$, $\alpha_i \in O$.
2. $Applicable(\overline{\alpha}, M_0)$.
3. $G \subseteq Result(\overline{\alpha}, M_0)$.

The last point implies that the goals in G have to be necessarily true in the last situation of the plan.

4.0.8 The Modal Truth Criterion

As there might be very many completions of a plan (in fact, the number might be infinite), calculating each completion every time we want to conclude something about the plan would be very inefficient. Therefore, Chapman presents a so-called modal truth criterion (MTC) for TWEAK plans[6], which is necessary and sufficient. The symbols p, q and r denote literals, s and t denote situations and C and W denote steps.

$$\Box\text{true}(p, s) \equiv \tag{56}$$

$$\begin{array}{l}
\exists t. \Box t \preceq s \wedge \Box\text{asserted-in}(p, t) \wedge \\
\forall C.\ \Box s \preceq C \vee \\
\quad \forall q.\ \Box\text{denies}(C, q) \Rightarrow \\
\quad\quad \Box(p \not\approx q) \vee \\
\quad\quad \exists W.\ \Box C \prec W \wedge \\
\quad\quad\quad \Box W \prec s \wedge \\
\quad\quad\quad \exists r. \Box\text{asserts}(W, r) \wedge \Box(p \approx q \Rightarrow p \approx r)
\end{array}$$

The situation t is the situation which makes p true; it is called the *establisher* of p. The establisher t can either be the initial situation or the output situation of some step. A step C which possibly denies p is called a

[6]The version presented here is slightly modified relative to Chapman's article; the motivation is presented in my thesis.

clobberer. For each clobberer, there must be a *white knight* W which comes between s and C and reasserts p whenever p is denied by C.

A subject of discussion is whether the truth criterion applies to *partial plans* or to complete and executable plans alone. Kambhampati and Nau [KN94] distinguishes between modal truth and *modal conditional truth*; the latter is independent of the executability of plans and therefore more suitable for partial plans. Modal conditional truth has a great resemblance to the transformation from $\mathcal{Q}$ to $\mathcal{K}$ discussed in a previous section of this report.

4.0.9 The TWEAK Algorithm

The heart of the TWEAK planning algorithm is the non-deterministic *goal-achievement procedure* (fig. 7), which is an algorithmic interpretation of the MTC. The procedure achieves a single literal goal by adding constraints and steps as necessary in order to make the MTC true for the literal in question. The top-level procedure of TWEAK is a loop where a goal is selected,

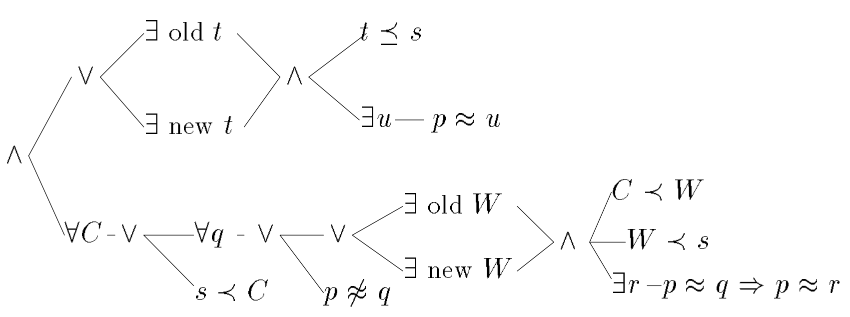

Figure 7: The goal-achievement procedure. The picture should be read from left to right. The symbols $\vee$ and $\exists$ represent nondeterministic choice-points; $\wedge$ denotes that all outgoing branches must be followed.

the goal-achievement procedure is called and the preconditions of any new steps become new goals. Chapmans suggests using a dependency-directed breadth-first search strategy (see e.g. [CM85]). As the algorithm is based on the necessary and sufficient MTC, it is correct and complete.

Theorem 5 (Correctness/Completeness) [Cha87] *If* TWEAK, *given a problem, terminates claiming a solution, the plan it produces does in fact solve the problem. If* TWEAK *returns signaling failure or does not halt, no solution exists.*

4.1 Epistemology and Ontology of TWEAK

The result of an unapplicable operator is not defined, so TWEAK belongs to the epistemological speciality $\mathcal{Q}_{\mathsf{p}}$.

There are no alternative results of actions and no local or structural ramifications. Each action has only one trajectory. Thus, TWEAK is completely within **I**. The actions have no duration and features are truth-valued, so TWEAK is **Isb**.

4.2 A Logicist Approach to TWEAK

Analogously with the STRIPS representation, we shall use an intermediate transformation from $\mathcal{Q}_{\mathsf{p}}$ to $\mathcal{K}_{\mathsf{p}}$.

The world and the plan is a FL chronicle

$$\langle \mathcal{K}, \mathcal{O}, \text{ELAW}, \text{SCD}, \text{OBS} \rangle. \tag{57}$$

A set of fixed atemporal constants denote the objects in the initial world and goal description. No domain closure is assumed. The atemporal constants not in this set are used as object variables.
The time domain is the non-negative integers $\mathbb{Z}$. A set of unfixed temporal constants $\{S_i\}_i$ represent the time-points of partially ordered actions.
ELAW represents the postconditions of actions in the form:

$$\mathsf{elaw} \quad [t, t'] \, A(\overline{x}) \leadsto [t, t'] \, p_1(\overline{x}_1) {:=} B_1 \wedge \ldots \wedge [t, t'] \, p_n(\overline{x}_n) {:=} B_n \tag{58}$$

where $p_i(\overline{x}_i)$ are atomic formulae and $B_i \in \{T, \ F\}$.
SCD is the plan; a set of actions of the form:

$$\mathsf{scd} \quad [S_i, S_i + 1] \, A_i(\overline{\boldsymbol{e}}_i) \tag{59}$$

where each S_i is a unique alphabetical temporal constant:

$$\mathsf{scd} \quad S_i \neq S_j \text{ for each } j \neq i \tag{60}$$

a partial order among actions (should be read "step S_i is before step S_j"):

$$\mathsf{scd} \quad S_i < S_j \tag{61}$$

and a set of codesignation constraints of the forms:

$$\begin{array}{ll} \mathsf{scd} & \boldsymbol{e}_i = \boldsymbol{e}_j \\ \mathsf{scd} & \boldsymbol{e}_i \neq \boldsymbol{e}_j \end{array} \tag{62}$$

where $\boldsymbol{e}_i, \ \boldsymbol{e}_j$ are atemporal terms.
OBS is the initial state; a set of formulas of the form:

$$\mathsf{obs} \quad [0] \, [\neg] p(\overline{\boldsymbol{e}}) \tag{63}$$

The goal and preconditions constitute a scenario description

$$\langle \mathcal{N}, \text{GOAL} \cup \text{QLAW}(\text{SCD}) \rangle \tag{64}$$

GOAL is the goals, a conjunction ($[\neg]$ denotes possible negation)

$$\mathsf{goal} \quad [S_0] \, [t, t'] \, [\neg] q_1(\overline{\boldsymbol{e}}_1) \wedge \ldots \wedge [t, t'] \, [\neg] q_n(\overline{\boldsymbol{e}}_n) \tag{65}$$

where $\overline{\boldsymbol{e}}_i$ are atemporal terms.
QLAW represents the preconditions of actions in the form:

$$\mathsf{qlaw} \quad [t, t'] \, A(\overline{x}) \leadsto [t, t'] \, [\neg] q_1(\overline{x}_1) \wedge \ldots \wedge [t, t'] \, [\neg] q_n(\overline{x}_n) \tag{66}$$

4.2.1 A Truth Criterion for TWEAK

Chapman's modal truth criterion can be reconstructed in a fairly straight-forward manner.

Theorem 6 (Truth criterion for non-linear plans) *Let* Γ *b a plan.*

$$\Gamma \mathrel{|\!\sim} \ulcorner[S_s]\,p(\overline{e}_s)\urcorner \quad \textit{iff} \tag{67}$$

$$\begin{array}{l}
\exists S_e\ [\ \Gamma \mathrel{|\!\sim} \ulcorner S_e < S_s\urcorner \wedge \Gamma \mathrel{|\!\sim} \ulcorner[S_e, S_e+1]\,p(\overline{e}_s){:=}T\urcorner \vee \\
\quad\quad \Gamma \mathrel{|\!\sim} \ulcorner[0]\,p(\overline{e}_s)\urcorner\ ] \wedge \\
\forall S_c\ [\ \Gamma \mathrel{|\!\sim} \ulcorner S_s \le S_c\urcorner \vee \\
\quad\quad \forall \overline{e}_c\ [\ \Gamma \mathrel{|\!\sim} \ulcorner[S_c, S_c+1]\,p(\overline{e}_c){:=}F\urcorner \Rightarrow \\
\quad\quad\quad\quad \Gamma \mathrel{|\!\sim} \ulcorner \overline{e}_s \neq \overline{e}_c\urcorner \vee \\
\quad\quad\quad\quad \exists S_w, \overline{e}_w\ [\ \Gamma \mathrel{|\!\sim} \ulcorner S_c < S_w\urcorner \wedge \\
\quad\quad\quad\quad\quad\quad \Gamma \mathrel{|\!\sim} \ulcorner S_w < S_s\urcorner \wedge \\
\quad\quad\quad\quad\quad\quad \Gamma \mathrel{|\!\sim} \ulcorner[S_w, S_w+1]\,p(\overline{e}_w){:=}T\urcorner \wedge \\
\quad\quad\quad\quad\quad\quad \Gamma \mathrel{|\!\sim} \ulcorner \overline{e}_c = \overline{e}_s \Rightarrow \overline{e}_w = \overline{e}_s\urcorner \\
\quad\quad\quad\quad]]]
\end{array}$$

For negated fluent statements, substitute $\ulcorner[..]\,\neg p(..)\urcorner$ *for* $\ulcorner[..]\,p(..)\urcorner$ *and* F *for* T *(and vice versa) above.*

4.2.2 An Algorithm for FL TWEAK

Just as for STRIPS, the adaptation of the algorithm is straight-forward. A goal-achievement procedure is obtained by interpreting the FL version of the truth criterion as an algorithm in the same way it was done for the original TWEAK. All choices are non-deterministic.

procedure goal-achievement($[S_s]\,p(\overline{e}_s)$)

1. Establishment — choose:
   (a) Infer that $\Gamma \mathrel{|\!\sim} \ulcorner[0]\,p(\overline{e}_e)\urcorner$, and infer or add:
   $$\textsf{scd}\quad \boldsymbol{e}_{si} = \boldsymbol{e}_{ei} \text{ for each } i \tag{68}$$
   (b) Select existing S_e such that $\Gamma \mathrel{|\!\sim} \ulcorner[S_e, S_e+1]\,p(\overline{e}_e){:=}T\urcorner$, and infer or add:
   $$\begin{array}{ll} \textsf{scd} & \boldsymbol{e}_{si} = \boldsymbol{e}_{ei} \text{ for each } i \\ \textsf{scd} & S_e < S_s \end{array} \tag{69}$$
   (c) Add a new step to Γ:
   $$\begin{array}{ll} \textsf{scd} & [S_e, S_e+1]\,A(\overline{e}') \\ \textsf{scd} & S_e < S_s \end{array} \tag{70}$$
   such that $\Gamma \mathrel{|\!\sim} \ulcorner[S_e, S_e+1]\,p(\overline{e}_s){:=}T\urcorner$.
2. Declobbering — for each existing S_c such that $\Gamma \mathrel{|\!\not\sim} \ulcorner S_s \le S_c\urcorner$ do
   (a) For all $\overline{e}_c$ such that $\Gamma \mathrel{|\!\sim} \ulcorner[S_c, S_c+1]\,p(\overline{e}_c){:=}F\urcorner$ choose:
      i. Promote the clobberer: $\textsf{scd}\ S_s < S_c$
      ii. Separate the steps: $\textsf{scd}\ \boldsymbol{e}_{si} \neq \boldsymbol{e}_{ci}$ for some i.

iii. White Knight — find an old step or add a new step $[S_w, S_w + 1]\, A(\overline{\boldsymbol{e}}')$ to Γ such that $\Gamma \mathrel{|\!\sim} \ulcorner[S_w, S_w + 1]\, p(\overline{\boldsymbol{e}}_w){:=}T\urcorner$ and add or infer:

$$\begin{array}{ll} \textsf{scd} & S_c < S_w \\ \textsf{scd} & S_w < S_s \end{array} \tag{71}$$

plus add atemporal constraints such that: $\Gamma \mathrel{|\!\sim} \ulcorner\overline{\boldsymbol{e}}_c = \overline{\boldsymbol{e}}_s \Rightarrow \overline{\boldsymbol{e}}_w = \overline{\boldsymbol{e}}_s\urcorner$

Observe that most suboperations of FL-TWEAK results in *monotonic extensions*.

4.2.3 An Example

We will illustrate the functionality of FL-TWEAK with a blocks-world example. We shall use the same world Γ_W as for the STRIPS example (fig. 3.4.3) in the previous section.

$$\begin{array}{ll}
\textsf{qlaw1} & [t,t']\, Pickup(x) \leadsto \\
& \quad [t]\, clear(x) \wedge [t]\, onTable(x) \wedge [t]\, handEmpty \\
\textsf{elaw1} & [t,t']\, Pickup(x) \leadsto \\
& \quad [t,t']\, clear(x){:=}F \wedge [t,t']\, onTable(x){:=}F \\
& \quad \wedge\, [t,t']\, handEmpty{:=}F \wedge [t']\, holding(x){:=}T \\
\textsf{qlaw2} & [t,t']\, Putdown(x) \leadsto [t]\, holding(x) \\
\textsf{elaw2} & [t,t']\, Putdown(x) \leadsto \\
& \quad [t,t']\, clear(x){:=}T \wedge [t,t']\, onTable(x){:=}T \\
& \quad \wedge\, [t,t']\, handEmpty{:=}T \wedge [t,t']\, holding(x){:=}F \\
\textsf{qlaw3} & [t,t']\, Unstack(x,y) \leadsto \\
& \quad [t]\, clear(x) \wedge [t]\, on(x,y) \wedge [t]\, handEmpty \\
\textsf{elaw3} & [t,t']\, Unstack(x,y) \leadsto \\
& \quad [t,t']\, clear(x){:=}F \wedge [t,t']\, clear(y){:=}T \\
& \quad \wedge\, [t,t']\, on(x,y){:=}F \wedge [t,t']\, handEmpty{:=}F \\
& \quad \wedge\, [t,t']\, holding(x){:=}T \\
\textsf{qlaw4} & [t,t']\, Stack(x,y) \leadsto [t]\, clear(y) \wedge [t]\, holding(x) \\
\textsf{elaw4} & [t,t']\, Stack(x,y) \leadsto \\
& \quad [t,t']\, clear(x){:=}T \wedge [t,t']\, clear(y){:=}F \\
& \quad \wedge\, [t,t']\, on(x,y){:=}T \wedge [t,t']\, handEmpty{:=}T \\
& \quad \wedge\, [t,t']\, holding(x){:=}F \\
\textsf{obs1} & [0]\, onTable(A) \wedge [0]\, on(B,A) \wedge [0]\, \neg clear(A) \\
& \wedge\, [0]\, clear(B) \wedge [0]\, handEmpty
\end{array} \tag{72}$$

The goal Γ_G is as follows:

$$\textsf{goal1} \quad [S_0]\, onTable(B) \wedge [S_0]\, on(A,B) \tag{73}$$

The planning process progresses as follows. For convenience, we will assume that the planner always makes the right choice at a non-deterministic choice-point. Fig 4.2.3 illustrates the status of the plan after some selected steps.

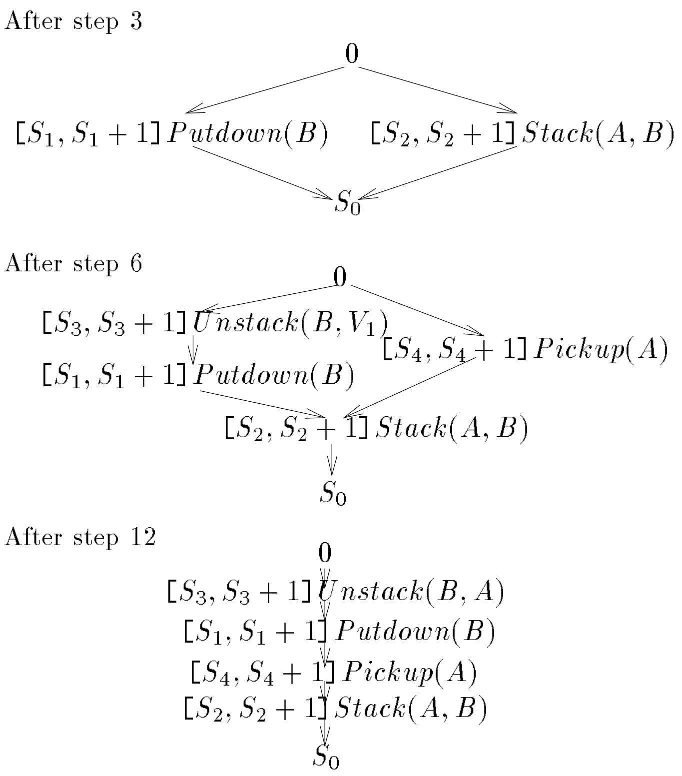

Figure 8: The status of the plan after some selected steps. The arrows represent temporal order.

1. Initially, the plan Γ' is empty.

2. TWEAK tries to achieve $[S_0]\, onTable(B)$.
   The goal achievement procedure suggests the following additions to Γ':

$$\begin{array}{ll} \textsf{scd1} & [S_1, S_1+1]\, Putdown(B) \\ \textsf{scd2} & S_1 < S_0 \end{array} \tag{74}$$

   New goals are $\{[S_1]\, holding(B)\}$.

3. TWEAK tries to achieve $[S_0]\, on(A, B)$.
   The goal achievement procedure suggests the following additions to Γ':

$$\begin{array}{ll} \textsf{scd3} & [S_2, S_2+1]\, Stack(A, B) \\ \textsf{scd4} & S_2 < S_0 \end{array} \tag{75}$$

   New goals are $\{[S_2]\, clear(B), [S_2]\, holding(A)\}$.

4. TWEAK tries to achieve $[S_1]\,holding(B)$.
The goal achievement procedure suggests the following additions to Γ':

$$\begin{array}{ll} \mathsf{scd5} & [S_3, S_3+1]\,Unstack(B, V_1) \\ \mathsf{scd6} & S_3 < S_1 \end{array} \tag{76}$$

An unfixed atemporal constant V_1 is added to represent the not yet known position of B at S_3.
New goals are $\{[S_3]\,clear(B),\ [S_3]\,on(B, V_1),\ [S_3]\,handEmpty\}$.

5. TWEAK tries to achieve $[S_2]\,clear(B)$.
The goal achievement procedure suggests the following additions to Γ':

$$\mathsf{scd7} \quad S_1 < S_2 \tag{77}$$

The already existing step $[S_1, S_1+1]\,Putdown(B)$ is used to achieve the goal.

6. TWEAK tries to achieve $[S_2]\,holding(A)$.
The goal achievement procedure suggests the following additions to Γ':

$$\begin{array}{ll} \mathsf{scd8} & [S_4, S_4+1]\,Pickup(A) \\ \mathsf{scd9} & S_4 < S_2 \end{array} \tag{78}$$

New goals are $\{[S_4]\,clear(A),\ [S_4]\,onTable(A),\ [S_4]\,handEmpty\}$.

7. TWEAK tries to achieve $[S_3]\,clear(B)$.
The goal is achieved by the inital state.

8. TWEAK tries to achieve $[S_3]\,on(B, V_1)$.
The goal achievement procedure suggests the following additions to Γ':

$$\mathsf{scd10} \quad V_1 = A \tag{79}$$

Thus the goal is satisfied by the initial state.

9. TWEAK tries to achieve $[S_3]\,handEmpty$.
The goal achievement procedure suggests the following additions to Γ':

$$\mathsf{scd11} \quad S_3 < S_4 \tag{80}$$

The goal is satisfied by the initial state, and the clobberer $[S_4, S_4+1]\,Pickup(A)$ is moved out of the way.

10. TWEAK tries to achieve $[S_4]\,clear(A)$.
The goal is achieved by $[S_3, S_3+1]\,Unstack(B, V_1)$, where $V_1 = A$.

11. TWEAK tries to achieve $[S_4]\,onTable(A)$. The goal is satisfied by the initial state.

12. TWEAK tries to achieve $[S_4]\,handEmpty$. The goal achievement procedure suggests the following additions to Γ':

$$\textsf{scd12} \quad S_1 < S_4 \tag{81}$$

    The goal is achieved by $[S_1, S_1 + 1]\,Putdown(B)$.

13. There are no goals left. If a previously achieved goal had been clobbered by a step addition, then this goal would have had to be reachieved. However, this is not the case for this planning problem.

The step order is $S_3 < S_1 < S_4 < S_2$, and the steps are as below (for clarity, we have substituted A for V_1 in scd5).

$$\begin{array}{ll} \textsf{scd5} & [S_3, S_3 + 1]\,Unstack(B, A) \\ \textsf{scd1} & [S_1, S_1 + 1]\,Putdown(B) \\ \textsf{scd8} & [S_4, S_4 + 1]\,Pickup(A) \\ \textsf{scd3} & [S_2, S_2 + 1]\,Stack(A, B). \end{array} \tag{82}$$

It is easily verified that the plan is executable and achieves the goal.

5 Summary

The topic of this report has been the planning problem from the perspective of Sandewall's *Feature and Fluents* framework.

First, we investigated how plans can be specified using *scenario descriptions*. We defined a nonmonotonic *fluent logic* for scenario descriptions. The fluent logic is based on second-order circumscription, where an *occlusion predicate* is minimized using a minimization policy called PMON. The resulting theories are reducible to first-order logic. Furthermore, the logic has the *restricted monotonicity property*; observations and temporal and atemporal constraints can be added with preserved monotonicity.

The fluent logic is used for representing plans and goals. We give extra consideration to the problem of reasoning about *partial plans* occurring in the intermediate steps of the planning process. Such plans usually contain unexecutable actions. The proposed solution is to separate the preconditions of an action from its effects.

After plan specification, we proceeded to plan synthesis. We analysed and reconstructed the total-order planner STRIPS within the Feature and Fluents framework. First, we identified the epistemological and ontological assumptions underlying the STRIPS formalism. A restricted case, called STRIPS-I, was found to belong to the $\mathcal{Q}_{\mathsf{pg}}$-**Isb** class (qualified action laws, inertia with single-step actions and binary feature values). Loosening the restrictions could lead to subcases of the ontological specialities alternative results **A** (but not context-dependency in this case), local ramification **U**, and structural ramification **D**. Second, we formulated an FL representation for STRIPS plans and stated a so-called truth criterion for this representation. Third, a recursive version of STRIPS was reconstructed to operate on this new representation.

We analysed and reconstructed the partial-order planner TWEAK in the same way. The TWEAK formalism was found to belong to the $\mathcal{Q}_{\mathsf{p}}$-**Isb** class (qualified action laws, inertia with single-step actions and binary feature values). An FL representation for TWEAK plans was formulated, and the so-called modal truth criterion was converted to this representation. The *goal achievement procedure* of TWEAK was reconstructed to operate on this new representation.

The original STRIPS formalism was constructed to side-step the frame problem; the solution was the use of add- and delete lists. In FL-STRIPS, the minimized occlusion predicate and the no-change premises Γ_{NCH} serve the same purpose, and this without introducing any additional complexity such as a large number of frame axioms.

The operators of the original STRIPS formalism are very restricted in their expressiveness. Basically the same kind of operators are used in many other planners, for instance TWEAK[Cha87], SNLP[MR91] and for primitive (non-decomposable) operators in hierarchical planners such as O-Plan[CT91] and SIPE[Wil88] (with the addition of resource specification). In contrast, FL is able to represent actions with duration, detailed trajectories, and context-dependent or random effects. If we are to construct planners for these specialities, we will only have to make minor modifications in our representation. As a matter of fact, extending STRIPS to deal with actions with duration seems to require moderate changes of the algorithm; in stead of adding actions $[\boldsymbol{t}, \boldsymbol{t}+1]\, A$, one simply adds $[\boldsymbol{t}, (\boldsymbol{t}+\boldsymbol{n})]\, A$. Research is currently going on to find logics for ramification and concurrency.

Finally, finding expressive and yet efficient languages for plan specification is not the only problem for agent planning. Other problems, which are generally outside the scope of traditional planning, concern knowledge acquisition, interference from external events and other agents, and the limited resources of reasoning and response times available to an agent in a real-life environment. Valuable work has been done in these areas [Moo85, BIP88, BD94] and we hope to explore them further in the future.

References

[AHT90] James Allen, James Hendler, and Austin Tate, editors. *Readings in Planning.* Morgan Kaufmann, 1990.

[All84] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984. Reprinted in [AHT90].

[All91] James F. Allen. Temporal reasoning and planning. In Allen, Kautz, Pelavin, and Tenenberg, editors, *Reasoning About Plans.* Morgan Kaufmann, 1991.

[BD94] Mark Boddy and Thomas Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67:245–285, 1994.

[BIP88] Michael E. Bratman, David J. Israel, and Martha E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349–355, 1988.

[BJ80] George S. Boolos and Richard C. Jeffrey. *Computability and Logic.* Cambridge University Press, 2nd edition, 1980.

[Bro91] Rodney A. Brooks. How to build complete creatures rather than isolated cognitive simulators. In Kurt VanLehn, editor, *Architectures for Intelligence*, pages 225–240. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1991.

[Cha87] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987. Reprinted in [AHT90].

[CM85] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence.* Addison Wesley, 1985.

[CT91] Ken Currie and Austin Tate. O-Plan: the open planning architecture. *Artificial Intelligence*, 52(1):49–86, 1991.

[DL94] Patrick Doherty and Witold Lukaszewicz. Circumscribing features and fluents. In *Proceedings of the 1st International Conference on Temporal Reasoning*, pages 82–100. Springer, 1994.

[Doh94] Patrick Doherty. Notes on PMON circumscription. Technical report, Department of Computer and Information Science, Linköping University, 1994.

[ENS92] Kutluhan Erol, Dana S. Nau, and V.S. Subrahmanian. When is planning decidable? In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, 1992.

[FN71] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971. Reprinted in [AHT90].

[GL87] M.P. Georgeff and A.L. Lansky, editors. *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop.* Morgan Kaufmann, 1987.

[GN87] Michael J. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, 1987.

[Gre69] Cordell Green. Applications of theorem proving to problem solving. In *Proceedings of the First International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, 1969. Reprinted in [AHT90].

[Gro88] Stephen Grossberg, editor. *Neural Networks and Natural Intelligence.* MIT Press, 1988.

[KN94] Subbarao Kambhampati and Dana S. Nau. On the nature and role of modal truth criteria in planning. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems.* Morgan Kaufmann, 1994.

[Kow77] Robert Kowalski. *Logic for Problem Solving.* Elsevier Science Publishing, New York, 1977.

[Lif85] Vladimir Lifschitz. Computing circumscription. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 121–127. Morgan Kaufmann, 1985.

[Lif87] Vladimir Lifschitz. On the semantics of STRIPS. In Georgeff and Lansky [GL87].

[Lif93] Vladimir Lifschitz. Restricted monotonicity. In *Proceedings of the Eleventh National Conference on Artificial Intelligence.* AAAI Press, Menlo Park, California, 1993.

[Luk90] Witold Lukaszewicz. *Non-monotonic Reasoning.* Morgan Kaufmann, 1990.

[McC62] John McCarthy. Situations, actions, and causal laws. Technical Report Memo nr 2, Stanford Artificial Intelligence Project, 1962.

[MH69] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969. Reprinted in [AHT90].

[Moo85] Robert C. Moore. A formal theory of knowledge and action. In J.R. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*, chapter 9. Ablex, Norwood, New Jersey, 1985.

[MR91] David McAllester and David Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence.* AAAI Press, Menlo Park, California, 1991.

[Nil80] Nils J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann, 1980. Also published by Springer-Verlag 1982.

[Ped86] Edwin P.D. Pednault. Formulating multiagent, dynamic-world problems in the classical planning framework. In Georgeff and Lansky [GL87].

[Ped88] Edwin P.D. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4:356–372, 1988.

[Ros81] Stanley J. Rosenschein. Plan synthesis: A logical perspective. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 331–37. Morgan Kaufmann, 1981. Reprinted in [AHT90].

[San94] Erik Sandewall. *Features and Fluents*. Oxford Press, 1994.

[SS94] Erik Sandewall and Yoav Shoham. Nonmonotonic temporal reasoning. In *Handbook of Artificial Intelligence and Logic Programming*. Oxford University Press, 1994. (To appear).

[Tat77] Austin Tate. Generating project networks. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 888–93. Morgan Kaufmann, 1977. Reprinted in [AHT90].

[Wil88] David E. Wilkins. *Practical Planning: extending the classical AI paradigm*. Morgan Kaufmann, 1988.