INFORMATICA, 1997, Vol. 8, No. 1, 3–22

PLAN GENERATION WITH THE LINEAR CONNECTION METHOD

Bertram FRONHÖFER

Institut für Informatik
TU München, D – 80290 München
E-mail: fronhoef@informatik.tu-muenchen.de

Abstract. The paper introduces to plan generation and the attempts and problems encountered with solutions by deductive methods. It presents the Linear Connection Method as a possibility to overcome the traditional shortcomings of logic for this application, which are discussed in great detail. This is followed by a guide to the research work carried out on the Linear Connection Method and related frameworks.

Key words: logics for plan generation, linear connection method, situation calculus, fluent calculus.

1. Introduction. How to reason about change is a classical issue of AI research, and generating plans is one of its particular topics (see the keyword 'planning' in Shapiro (1992)). Already in the late sixties/early seventies plan generation was conceived as an inference problem formulated in classical logic – the Situation Calculus – and attempts were made to automate plan generation by means of theorem proving. The nonsuccess of this endeavour turned the development of plan generation approaches and systems away from deductive systems.

In 1986 a new, logic-based approach – the Linear Connection Method – was proposed by W. Bibel (1986), which promised to overcome the drawbacks of the Situation Calculus. This approach has stimulated a considerable amount of further research over the last 10 years and is still in full evolution.

The plan of the paper is as follows. In Section 2 we define plan generation and related problems like plan verification, reactive systems, etc. We next discuss Situation Calculus and its drawbacks, and then introduce the Linear Connection Method together with a simple proof search algorithm (Linear Backward Chaining). Section 3 is devoted to a comparative analysis of Linear

Backward Chaining and proof search in the Situation Calculus. In Section 4 we finally review more advanced and related research work. We survey investigations into the logics/semantics of the Linear Connection Method and related approaches, list extensions to overcome limitations of the original framework, and point to further implementations and more refined proof/plan search algorithms. Finally, we also refer to applications of the Linear Connection Method to domains which are different from plan generation.

2. Logic and plan generation. Problems dealing with reasoning about change are generally composed of some of the following four essential components:

- A *set of actions*, where an action is a description of a particular type of (destructive) change. In most general, terms an action must have *preconditions* which decide its applicability and *effects* which result from its application. The effects are subdivided into facts which are created (*added*) and facts which are destroyed (*deleted*) by the action.
- A description of a state of the current world – the *initial situation.*
- A (maybe partial) description of a desired future state – the goal.
- A (partially or totally ordered) set of (instantiated) occurrences of actions – a *plan.*

There are the following, basically different, types of problems which can be created from the four components just mentioned.

- *Reactive Systems.* Here we are given a current (initial) situation and a set of actions. All actions which are applicable to the current situation compete for execution. The application of the winning action generates a new situation where the competition between the now applicable actions starts again. (Note that the actions are not put under any external control which composes purposeful plans).
- *Plan Evaluation, Temporal Projection and Plan Verification.* Here we are given a plan, and we ask whether this plan is applicable to an initial situation and produces certain wanted effects (plan evaluation), or we ask how does the world look like after its successful application (temporal projection). More generally, we may be interested in verifying certain properties of the given plan, e.g., whether it satisfies certain purposes, obeys certain restrictions or constraints, etc., which requires to *reason about actions.*

- (*Goal-Oriented*) *Plan Generation.* Here we are given a current (initial) situation, a set of actions and a goal, and we ask whether there is a plan which transforms the initial situation into a new situation in which the goal is satisfied. (Note that in contrast to reactive systems, the actions don't act 'on their own', but their composition is controlled by a plan search procedure).

These types of problems can be understood as deductive ones and for plan generation – the one we will focus on in the sequel – we would get the inference problem to prove a 'specification theorem' of the form:

$$\textit{Initial Situation} \text{ and } \textit{Actions} \text{ imply } \textit{Goal}$$

from whose proof a plan will be extracted. This informal formulation gives rise to the question of a suitable logic which allows both a convenient specification of plan generation problems and, even more importantly, an efficient automated proof search.

2.1. Situation calculus. The first proposal how to generate plans via theorem proving was made on the choice of classical first-order logic and resulted in the so-called Situation Calculus (see McCarthy and Hayes (1969)), which immediately fell into disrepute due to bad practical performance in attempts to prove the respective specification theorems by use of automated theorem provers (see Green (1957)).

Since facts may change their truth value over time, with Situation Calculus a 'situation argument' was added to each predicate, e.g., $on(a, b, s)$ says that the block a is on top of block b in situation s. For facts being true in the initial situation, the situation argument is a reference to this situation, and the goal is described by a formula with an existentially quantified variable as the situation argument. The actions are of the form: if certain facts are true in a situation x, then in the situation $applied(a, x)$ – resulting from the application of action a to situation x – some other facts become true (see Nilsson (1982) for a good introductory presentation).

Unfortunately, we need additional assertions – so-called *Frame Axioms* – which state for every fact which is not affected by action a, that it will still be valid in situation $applied(a, x)$, if it was valid in situation x. Since, in general, an action changes only very few facts, the number of Frame Axioms is close to $|set\ of\ facts| \times |set\ of\ actions|$. This huge amount of Frame Axioms

was considered the culprit – commonly referred to as the *Frame Problem* – for the inadequateness of Situation Calculus[1] . Since this problem is also inherent in other logics (e.g., modal logics) known in those days, the nonsuccess of Situation Calculus discredited the use of logic for plan generation in general, and henceforth planning systems were conceived without reference to a particular kind of logic [2].

2.2. Linear connection proofs. In Bibel (1986), a new logic-oriented approach – called the *Linear Connection Method* – has been proposed whose great advantage was to work without Frame Axioms. These were not needed because in contrast to Situation Calculus, no 'situation information' is encoded in every fact which may vary over time. Thus, it promised to overcome the Frame Problem.

Instead of $on(a, b, s)$ we get just $on(a, b)$, and the initial situation, as well as the goal, are conjunctions of such literals. An action is a formula (*action implication*) of the form

$$A_1 \wedge \ldots \wedge A_g \rightarrow C_1 \wedge \ldots \wedge C_h$$

with facts $A_1, \ldots A_g$, $C_1, \ldots, C_h$ (without situation arguments).

Of course, we cannot expect every (classical) proof of a specification theorem given this way to yield a correct plan. Since, unlike to Situation Calculus, our literals are not explicitly time-dependent, they can be reused again and again, although they should no longer be 'valid' after the execution of certain actions. However, a notable feature of intuitively intended proofs of specification theorems made with the Connection Method (Bibel, 1987) is that every

[1] Although identified in the context of Situation Calculus as a very special technical problem, it soon turned out that the Frame Problem in the sense of "how does the 'frame' change" when an action is executed, has a much larger scope of importance. However, in the following our attention will be focused on the purely technical or inferencial Frame Problem of Situation Calculus.

[2] Let us shortly mention that the rupture between logic and planning was mainly on deductive grounds, since, for many planning systems, efforts were made to define precise declarative semantics. A good example in this respect is the well-known STRIPS approach (Fikes and Nilsson, 1971) for which semantics were given in Lifshitz (1986), but STRIPS's connectives and a proof theory were never worked out.

instance of a literal is used at most once, i.e., it is involved in at most one connection. Proofs of that kind are called *Linear Connection Proofs* and it is claimed in Bibel (1986) that this kind of '*linearity*' is the necessary restriction to be imposed on proofs in order to generate correct plans.

Apart from the mentioned linearity, the Linear Connection Method demands the following *non-formal/semantic requirements* about the intended meaning of the action implications to be obeyed. The *antecedent* $A_1 \wedge \ldots \wedge A_g$ must comprise all facts of the existing situation which are involved in the action – either as being necessary conditions for the action's application or as being facts which shall no longer be valid after the action has been carried out. The *consequent* $C_1 \wedge \ldots \wedge C_h$ must comprise all facts which are either newly created by the action or which were involved in the action as preconditions, but are not affected by it. This convention about the specification of actions entails that all those facts of a situation, which are not included in the antecedent shall survive the application of the action; a property which harmonizes perfectly with the working of Linear Connection Proofs. This harmony between specification philosophy and formal proof concept is the ultimate reason why no Frame Axioms need to be given with this approach.

2.3. Linear backward chaining. That the Linear Connection Method outperforms the Situation Calculus can be seen easily with the straightforward proof search algorithm called *Linear Backward Chaining* (LBC) – presented below – which virtually constructs Linear Connection Proofs by backward search from the goal clause rather analogously to the ordinary top-down evaluation of PROLOG.

The main difference to PROLOG is the treatment of unit clauses (facts) which may be used at most once.

A further decisive point is the transformation of an action implication

$$A_1 \wedge \ldots \wedge A_g \rightarrow C_1 \wedge \ldots \wedge C_h$$

into the following h rules

$$\begin{aligned}
C_1 \;&:-\; A_1, \ldots, A_g,\ \texttt{NewFact}(C_2), \ldots, \texttt{NewFact}(C_h)\\
C_2 \;&:-\; A_1, \ldots, A_g,\ \texttt{NewFact}(C_1),\ \texttt{NewFact}(C_3), \ldots, \texttt{NewFact}(C_h)\\
&\ \ \vdots\\
C_{h-1} &:-\; A_1, \ldots, A_g,\ \texttt{NewFact}(C_1), \ldots \texttt{NewFact}(C_{h-2}),\ \texttt{NewFact}(C_h)
\end{aligned}$$

$C_h \quad :- \quad A_1, \ldots, A_g,\ \texttt{NewFact}(C_1), \ldots \texttt{NewFact}(C_{h-1})$

which make use of a special built-in predicate `NewFact` which stores created facts. This set of stored facts is initialized by the facts of the initial situation, and a fact A is removed when it unifies with a subgoal A in one of the selected rules, and a fact C is added when a call of the subgoal $\texttt{NewFact}(C)$ is evaluated. (Of course, both removing and adding of facts must be backtrackable).

That the LBC algorithm is a correct and complete proof procedure for the Linear Connection Method has been shown in Fronhöfer (1996a), (see also Fronhöfer (1996e) for a comparison of the LBC algorithm with Situation Calculus and with the planning system UCPOP (Penberthy and Weld, 1992) by means of evaluations on benchmarks).

3. Comparative proof search behaviour. Before we analyse the differences in proof search between the LBC procedure and Situation Calculus, we will first have a closer look at the Frame Problem of the Situation Calculus. This technical or inferencial Frame Problem seems to have three aspects which have stimulated different kinds of research work.

- *Size of specifications.* It is certainly a nuisance of Situation Calculus that lots of axioms must be written down which serve no other purpose than to express that most facts of a situation are not at all affected by a certain action. (Apart from being boring work to do, it is also extremely prone to error and the specifications tend to be extremely large.) A first proposal to reduce the effort to be spent on writing Frame Axioms was made by Kowalski. He turned facts into terms – thus allowing to quantify over 'fact variables' – which permitted to reduce the Frame Axioms to one per action by just listing up the literals which are 'exempted from survival' (see Kowalski (1979) or Nilsson (1982)). Similar ways to compress sets of Frame Axioms into smaller formulae are found in Schubert (1990). A further more convenient proposal has been made by Reiter (1991) where Frame Axioms can be generated from stated properties about predicates and actions.
- *Size of proofs.* The Frame Axioms increase the size of proofs by $O(m)$ where m is the size of the initial situation (see Hölldobler and Thielscher (1996)).
- *Size of search spaces.* Last, but certainly not least, Frame Axioms may tremendously blow up the search space in case of a backward plan/proof

search. We consider forward chaining systems less suitable, because of the risk that large initial situations might slow down proof search tremendously, because lots of actions might be applicable to the initial situation without any orientation towards the goal to achieve.) The mentioned methods proposed by Kowalski and Reiter just reduce the size of the specification, but generate the same search spaces since they get all the original Frame Axioms back as instances or as consequences of their generative processes; thus they are of no help for proof search.

Of course all these problems deserve attention, however, we consider the third one to be ultimately crucial. Convenient ways to specify plan generation problems don't help if the resulting specifications are not computationally tractable, and although it is nice to get small proofs, the lesson learnt from classical theorem proving is to be happy if any proof is found at all. But for finding a proof the size of the search space is decisive, for which reason we will have now a closer look at.

Although the Frame Problem is the most widely known shortcoming of Situation Calculus – which might be particularly due to the effort of specification it entails, and which is already extremely shocking for the layman who has little background in logic and theorem proving – it is by far not the only drawback of Situation Calculus.

All together we identified the following trouble spots of the search space of Situation Calculus (in case of backward proof search), which can all be overcome with the LBC procedure.

- *Frame problem.* Given an action system consisting of n actions and let us consider a certain fact F. Then there will be some actions which create F, some others which destroy F and all the remaining actions – let us assume that there are $k \leqslant n$ of them – will leave F untouched. In particular, in large sets of actions k may be rather close to n.) If we now want to prove F we have – apart from the actions which create F – also k Frame Axioms which allow us to derive F, which yields an additional branching factor of k. In case of a proof which shall produce a plan consisting of l actions this results in an additional search space of $\sum_{i=1}^{l} k^i$, i.e., we get a practically intractable search space already for simple, but not completely trivial plan generation problems.

The absence of Frame Axioms in the case of Linear Connection Proofs reduces k to 0 [3].

- *Shared situation variables.* In general, a goal formula consists of more than one literal; the same holds for the preconditions of actions. For two literals A and B we obtain with Situation Calculus, the problem to prove: $\exists Z : A(Z) \wedge B(Z)$ where Z will refer to a plan (encoded as a term) which leads to a situation in which both A and B hold. The shared variable Z may cause a lot of backtracking if the value/plan obtained as the result of a proof of $A(Z)$ is not suitable for B. This kind of backtracking will not occur when working with Linear Connection Proofs due to the absence of such variables. Note however, that we are still not able to work with completely independent proofs for A and B which can be combined into a proof of $A \wedge B$ as in classical logic, but our algorithm tries to achieve B with a plan P_B which is a continuation of the plan P_A which produced A. Only if B cannot be generated by a plan which is a continuation of P_A, the algorithm backtracks.
- Repeated subcomputations. Whenever an action a produces more than one single new fact – e.g., we have a formula of the form $\forall S : A(S) \rightarrow B(apply(a, S)) \wedge C(apply(a, S))$ – a lot of time may be spent on repeated computations of the same subgoals. Imagine we want to prove $\exists Z : B(Z) \wedge C(Z)$ then we first show $B(Z)$ for which we have to solve $A(S)$, and afterwards show $C(Z)$ which requires to solve $A(S)$ again. The LBC algorithm needs not to do a recomputation at all, because after the proof of B, the literal C will be memorized via the `NewFact`-mechanism.[4]

Considering these issues in view of the different kinds of problems stated in Section 2 we come to the conclusion that only plan generation (with backward search) is really affected by the search space explosion problem of Situation Calculus, and thus the only one of these problems for which the development of

[3] Recent experiments exploiting disequality constraints achieved just a reduction of k to 1 for Situation Calculus (see Fronhöfer (1996e)).

[4] Attempts to avoid recomputations through the use of a lemma facility were rather discouraging due to the increase of the search space caused by the generated lemmata. The `NewFact`-mechanism instead seems to be more a fine-grained lemma handling and to be more tuned to the application.

proof procedures based on the Linear Connection Method promises to be a big step forward. Optimal control of proof/plan search by forward chaining would apply Frame Axioms only after the application of an action in order to compute the resulting situation. This means an increase of inferences which is linear in the size of the situations. We get the same increase in case of a reactive system. If a plan is already given, the shared variables cause no backtracking, but just failure, and first experiments indicate that avoiding repeated subcomputations by lemma facilities in this case seems to speed up rather than to slow down the proof search.

4. Survey of further research. After having introduced in the preceding sections the basic issues and ideas, we want to devote the rest of the paper to an overview of the research activities in this field. We will outline the principal research directions following a rather chronological order of presentation.

As already mentioned, Linear Connection Proofs were first proposed in Bibel (1986). This paper presents and motivates the underlying idea and shows its intuitive usefulness by means of several examples.

This article also raised many open questions and in Fronhöfer (1987a) a first analysis of the concept of Linear Connection Proofs was carried out. Its scope was fairly restricted to questions of proof procedures and especially to the workings of the Connection Method under the assumption of linearity.

4.1. Logical investigations. Among the first questions raised by Linear Connection Proofs, the most important one concerned the logic underlying this approach, or as often stated, the question of its semantics. In Bibel *et al.* (1989) three semantics for Linear Connection Proofs were presented. A first one was inherited from classical logic through an embedding of Linear Connection Proofs into the Situation Calculus, a second one was a possible world semantics which was directly attached to the language of Linear Connection Proofs, while a third semantics exploited the idea of 'rewriting' sets of literals.

Towards the end of the eighties, Linear Connection Proofs were reformulated on the basis of equational logic programming (ELP) – see Große *et al.* (1992b) (also available as Große *et al.* (1992b)) or Schneeberger (1992). Hölldobler (1996) gives an introduction to this approach and an overview of further developments based on this framework. (Recently, the name Fluent Calculus became popular for this approach and we will use it in the following.) In this

framework a state of the world is encoded as a term built up with a binary function symbol $\circ$ which models a (multiplicative) conjunction. Facts become simple terms with constants as arguments. An action is represented as a pair of $\circ$-terms – which together with the action's name yields a ternary predicate – and an action application is implemented via commutative-associative (non-idempotent) unification (see Thielscher (1992); a short English summary is found in Große *et al.* (1992b)). Thus, an embedding into classical logic has been achieved from where the semantics can be inherited.

A further contribution to the discussion of the underlying logic was the embedding of Linear Connection Proofs into the modal logic $K4$ in Fronhöfer (1991a). Via this embedding a semantics for Linear Connection Proofs could simply be inherited from modal logic: states in planning were identified with possible worlds.

The advent of Girard's Linear Logic (Girard, 1987) stimulated a further line of research. In Masseron *et al.* (1993) (see also Masseron *et al.* (1990)) a reconstruction of Linear Connection Proofs on the basis of Linear Logic was proposed. In this approach, plan generation was not conceived as proving a theorem (from logical axioms), but the initial state and the actions are stated as nonlogical axioms (sequents) to which the inference rules of sequent calculus for Linear Logic are applied in order to derive future states of the world.

This work was studied and analyzed in Fronhöfer (1992) and Große *et al.* (1996a) (the latter paper going back to the earlier research report Große *et al.* (1992a)). These investigations showed that apart from minor differences, the Linear Connection Method, the ELP-Framework and the approach of Masseron *et al.* (1993) are equivalent.

An alternative way to Masseron *et al.* (1993) has been pursued in Fronhöfer (1996a), where specification theorems (of plan generation problems) with a Linear Connection Proof are characterised as derivable in a (multiplicative) sequent system which differs from classical logic just in a restriction of contraction to implications.

Recently, in Bartheye (1994) the Linear Logic based reconstruction of Linear Connection Proofs was transformed into a plan calculus based on algebraic notions.

Further work which must be mentioned in this context are Thielscher (1994a) and Thielscher (1994b) where translations between the Fluent Calculus and the

action description language $\mathcal{A}$ (Gelfond and Lifshitz, 1993), and between $\mathcal{A}$ and the Ego-World-Semantics (Sandewall, 1994) are worked out, which allows to inherit the semantics from these approaches.

4.2. Extensions. As already mentioned, the original proposal of Linear Connection Proofs was limited to a strips-like action language. To overcome this limitation lots of investigations into various directions have been carried out.

In Fronhöfer (1991a) an extension of the action format of Linear Connection Proofs through *subimplications* (subactions) was proposed. These subimplications – whose execution is obligatory whenever possible – allow to specify 'side-effects' of an action, e.g., in case of carrying around a briefcase – the main action – the subaction would specify the change of location of objects which are in the briefcase.

A very similar proposal – called *specificity* – for tackling these and related problems is found in Hölldobler and Thielscher (1993b), Hölldobler and Thielscher (1993a) and Hölldobler and Thielscher (1995) where actions are particularised in different degrees of specialisation, and the proof search process is obliged to always choose the most specific one which is applicable. For example, in case of dealing with fragile objects, we would get two formulations of the action of dropping an object. A general one and a more special one which states that an object will break if it is fragile. The latter one has to be chosen whenever possible, i.e., whenever a fragile object is dropped.

A further important contribution is the extension through (additive) *disjunction* (see Brüning *et al.* (1993); Brüning *et al.* (1992) or Brüning *et al.* (1994)) in order to model actions with alternative undetermined results. This allows, for instance, to express actions with disjunctive results like throwing a coin.

In Thielscher (1995) and Thielscher (1997) the Fluent Calculus is extended to cope with *ramifications*, i.e., with indirect effects of actions. Apart from the specification of actions, additional domain constraints and causal relationships are stated. In this setting, the application of an action is followed by the application of causal relationships until the underlying domain constraints are satisfied. A further extension to cope with the *qualification problem*, i.e., to assume away abnormal disqualifications for actions, is developed in Thielscher (1996).

In Bornscheuer and Thielscher (1994) and Bornscheuer and Thielscher (1996) a sound and complete encoding of the language $\mathcal{A}_C$ (Baral and Gelfond

(1993) – an extension of the action description language $\mathcal{A}$, which supports the description of concurrent actions – into the Fluent Calculus is presented. Moreover, a further extension called $\mathcal{A}_C^+$ is proposed, which allows to infer sound information from contradictory descriptions and to describe non-determinism and uncertainty. Finally, the encoding of $\mathcal{A}_C$ into the Fluent Calculus is extended to $\mathcal{A}_C^+$. Bornscheuer and Thielscher (1997) deals with nondeterministic actions and the view of uncertain and contradictory knowledge as explicit resp. implicit indeterminism.

In Herrmann and Thielscher (1996b) a new proposal is made for handling time in case of continuous actions. Instead of discretizing into time slices of equal length, a separation of time into slices of varying length was worked out. The resulting framework – a combination of deduction and numerical calculus – was embedded into the Fluent Calculus, which is described in Herrmann and Thielscher, (1996a).

Quite a general problem is the detection of unsolvable planning problems, for which a solution is proposed in de Waal and Thielscher (1995) through the exploitation of techniques from program analysis and transformation.

4.3. Algorithms and implementations. Up to now several attempts have been made to implement Linear Connection Proofs or one of the related systems.

- The first prototypical implementation in PROLOG was made at the TH Darmstadt (see Hölldobler and Schneeberger (1990) or Große *et al.* (1992b) for a detailed description). It is based on the ELP formalism. The system can do both forward and backward search, although forward searching is preferred due to certain extensions as specificity whose application is cumbersome when searching backward.
- A further PROLOG based implementation was made by Eric Jacopin at LAFORIA, Paris. Inspired by the work of Masseron *et al.* (1993), a proof search algorithm which constructs sequent calculus derivations, was worked out and implemented (see Jacopin (1993a) and Jacopin (1993b)). To reduce the search space, various efforts were made to restrict the applicability of some of the sequent rules – without sacrificing completeness, of course.
- A third experimental prototype has been implemented on the basis of the LBC algorithm presented before (see Fronhöfer (1996e) and shorter in Fronhöfer (1996a)).

All these systems are just first straightforward prototypes, which need to be further developed.

Proposals for increasing the efficiency of a planner, like the ones just mentioned by means of lemmata or through loop prevention, are found in Brüning (1993).

Further improvements of the LBC algorithm are described in Fronhöfer (1996b) (excerpts of this report are published in Fronhöfer (1996d) and Fronhöfer (1996c)). The starting point is that action implications like $A \wedge B \rightarrow A \wedge D$ yield a cyclic rule $\underline{A}$:- $\underline{A}, B, \texttt{NewFact}(D)$ with the transformation given in Subsection 2.3. Although such cyclic rules look tautological, they are indispensable as can be seen with the plan search for the so-called Sussman Anomaly (see Nilsson (1982)). On the other hand, they look a bit like Frame Axioms, and pros and cons of this view are discussed in Fronhöfer (1996d). In any case, they are troublesome from the point of view of plan search, because they may cause looping, and fortunately, we can get rid of them by allowing to insert actions into an already constructed partial plan: the resulting so-called LIP-algorithm and an experimental evaluation of the speed up which can be achieved this way are found in Fronhöfer (1996c). This LIP-algorithm can also be seen as a partial order planning method for Linear Connection Proofs.

A further partial order algorithm – on the basis of the Fluent Calculus – is presented in Hölldobler and Schneeberger (1996). Here the stimulus was not the analysis of inefficiencies of an existing proof procedure, but direct orientation at the work on partial order planning. The idea is to allow arbitrary subgoal selection (in the subgoal stack) and to record constraints about the necessary temporal ordering of introduced actions.

A completely different investigation is Eder *et al.* (1996), where based on the Chemical Machine (Berry and Boudol, 1990) – a general model for parallel computation – a computational model for the Fluent Calculus is developed.

4.4. Further applications. Apart from planning – its primary application – also other possibilities to make use of Linear Connection Proofs were pursued.

Early attempts were investigations into the applicability of Linear Connection Proofs for a combination of logical and procedural programming (see Fronhöfer (1987b); Fronhöfer (1992) and Fronhöfer (1988)) exploiting on the one hand the proximity to logic and on the other hand the possibility to change the contents of memory due to the availability of actions.

In Fronhöfer (1991c) Linear Connection Proofs were applied to the modeling of inheritance with exceptions based on the view of concepts as sets of attributes and roles. Passing from a concept to a subconcept would be modeled as an action which in case of an exception deletes the respective attribute(s).

A completely different, but somehow related approach – using Linear Logic – to exception handling is proposed in Vauzeilles and Fouqueré (1993) which is based on reformulation of is-a-nets.

In this context also the following work should be mentioned: Linear Logic based approaches to updating in Girard (1990), and to object oriented programming, e.g., the system Linear Objects by R. Pareschi and J.M. Andreoli (Andreoli and Pareschi, 1991). In Sigmund (1992) object-oriented programming à la Linear Objects was remodeled in the setting of Fluent Calculus.

A further, quite different application of Linear Connection Proofs, was their use in Fronhöfer (1991b) for modeling a system for generating cooperative answers. Understanding a query as a a partial goal state, the plan constructed while answering it, produces further facts of the goal state, which constitute additional information to be given to the user.

In Thielscher and Schaub (1995) planning techniques are applied to nonmonotonic reasoning. The task of credulous reasoning in default logic is reformulated as a deductive planning problem in the setting of Fluent Calculus. Thus a proof procedure for credulous reasoning is inherited from planning and also theoretical (complexity) results from planning could be transferred to default reasoning.

Acknowledgements. This paper has its origin in a tutorial given by the author at the TEMPUS Summer School, Vilnius, organized by Vytautas Čyras, whom we owe thanks for the invitation. Further thanks to Michael Thielscher who commented on a draft of this paper, and also thanks to Joachim Steinbach for a critical reading in the end.

REFERENCES

Andreoli, J.-M, and R. Pareschi (1991). Linear objects: Logical processes with built-in inheritance. *New Generation Computing*, **9**(3+4), 445–474.

Baral, C, and M. Gelfond (1993). Representing concurrent cctions in extended logic

programming. *IJCAI-93,* 866–871.

Bartheye, O. (1994). *Calcul de plans d'action: des méthodes déductives vers les méthodes algébriques.* PhD thesis, Université d'Aix-Marseilles II, Faculté des Sciences de Luminy.

Berry, G., and G. Boudol (1990). The chemical abstract machine. In *ACM Symposium on Principles of Programming Languages.* pp. 81–94.

Bibel, W. (1986). A Deductive solution for plan generation. *New Generation Computing,* **6**, 115–132.

Bibel, W. (1987). *Automated Theorem Proving.* Vieweg (Second Edition).

Bibel, W., L. Fariñas del Cerro, B. Fronhöfer and A. Herzig (1989). Plan generation by linear proofs: on semantics. In D. Metzing (Ed.), *GWAI'89, 13th German Workshop on Artificial Intelligence.* Informatik-Fachberichte 216, Springer, Schloß Eringerfeld, Geseke. pp. 49–62.

Bornscheuer, S.-E, and M. Thielscher (1994). Representing concurrent actions and solving conflicts. In B. Nebel and L. Dreschler-Fischer (Eds.), *Proceedings of the German Annual Conference on Artificial Intelligence (KI).* LNAI 861. Springer, Saarbrücken. pp. 16–27.

Bornscheuer, S.-E, and M. Thielscher (1996). Representing concurrent actions and solving conflicts. *Journal of the IGPL,* **4**(3), 355–368.

Bornscheuer, S.E., and M. Thielscher (1997). Explicit and implicit indeterminism: Reasoning about uncertain and contradictory specifications of dynamic systems. *Journal of Logic Programming,* Special Issue 'Action and Change' (to appear).

Brüning, S., G. Große, S. Hölldobler, J. Schneeberger, U. Sigmund and M. Thielscher (1993). Disjunction in plan generation by equational logic programming. In A. Horz (Ed.), *Beiträge zum 7. Workshop Planen und Konfigurieren.* Arbeitspapiere der GMD 723.

Brüning, S. (1993). Towards efficient calculi for resource oriented linear deductive planning. *Technical Report AIDA–93–17,* FG Intellektik, FB Informatik, TH Darmstadt.

Brüning, S., G. Große, S. Hölldobler, J. Schneeberger, U. Sigmund and M. Thielscher (1992). On disjunction in linear logic programming (extended abstract). In D. Miller (Ed.), *Proceedings of the Workshop on Linear Logic and Logic Programming.* Report MS–CIS–92–80. Univ. of Pennsylvania, School of Engineering and Applied Science, Computer and Information Science Department. pp. 53–59.

Brüning, S., S. Hölldobler, J. Schneeberger, U. Sigmund and M. Thielscher (1993). Disjunction in resource-oriented deductive planning. In D. Miller (Ed.), *Proceedings of the Int. Logic Programming Symposium.*

Brüning, S., S. Hölldobler, J. Schneeberger, U. Sigmund and M. Thielscher (1994). Disjunction in resource-oriented deductive planning. *Technical Report AIDA–94–03,* FG Intellektik, FB Informatik, TH Darmstadt.

de Waal, A., and M. Thielscher (1995). Solving deductive planning problems using program analysis and transformation. In M. Proietti (Ed.), *Proceedings of the Int.*

Workshop on Logic Program Synthesis and Transformation (LOPSTR), Vol. LNCS 1048. Springer, pp. 189–203.

Eder, K., S. Hölldobler and M. Thielscher (1996). An abstract machine for reasoning about situations, actions, and causality. In R. Dyckhoff, H. Herre and P. Schroeder-Heister (Eds.), *Proceedings of the Int. Workshop on Extensions of Logic Programming,* Vol. LNAI 1050. Springer pp. 137–151.

Fikes, R., and N. Nilsson (1971). STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence,* **2**, 189–208.

Fronhöfer, B. (1992). Planlog. In S.C. Shapiro (Ed.), *Encyclopedia of Artificial Intelligence.* J. Wiley & Sons.

Fronhöfer, B. (1987a). Linearity and plan generation. *New Generation Computing,* **5**, 213–225.

Fronhöfer, B. (1987b). Planlog: A language framework for the integration of logical and procedural programming. In J. McDermott (Ed.), *IJCAI –87.* Kaufmann Inc., Los Altos. pp. 15–17.

Fronhöfer, B. (1988b). Plan schemes in planlog. In T.O'Shea and V. Sgurev (Eds.), *AIMSA-88, Artificial Intelligence – Methodology Systems Applications.* North–Holland, Amsterdam pp. 169–176.

Fronhöfer, B. (1991a). Default connections a modal planning framework. In J. Hertzberg (Ed.), *European Workshop on Planning (EWSP–91).* LNAI 522. Springer, Bonn. pp. 39–52.

Fronhöfer, B. (1991b). Generating cooperative answers with a transition framework. In R. Demolombe, L.F. del Cerro and T. Imielinski (Eds.), *Nonstandard Queries and Answers,* Vol. 2. ONERA_CERT, Toulouse pp. 127–144.

Fronhöfer, B. (1991c). Implementing exceptions in inheritance by concept transforming actions. In E. Ardizzone, S. Gaglio and F. Sorbello (Eds.), *Trends in Artificial Inteligence,* LNAI 549. Springer, Palermo. pp. 58–67.

Fronhöfer, B. (1992). Linear proofs and linear logic. In D. Pearce and G. Wagner (Eds.), *Logics in AI.* JELIA'92 LNCS 633. Springer, Berlin. pp. 106–125.

Fronhöfer, B. (1996a). *The Action-as-Implication Paradigm: Formal Systems and Application,* Vol. 1 of *Computer Science Monographs.* CSpress, München. (revised version of Habilitationsschrift, TU München 1994.)

Fronhöfer, B. (1996b). Cutting connections in linear connection proofs. *Technical Report AR-96-01,* Technische Universität München, available from ftp://ftp.informatik.tu-muenchen.de/local/lehrstuhl/jessen/Automated_Reasoning/Reports/AR-96-01.ps.gz.

Fronhöfer, B. (1996c). Cutting connections in linear connection proofs. In *Int. Computer Symposium'96,* Kaohsiung, Taiwan. Sun Yat–Sen University. pp. 109–116.

Fronhöfer, B. (1996d). Cyclic rules in linear connection proofs. In G. Görz and S. Hölldobler (Eds.), *KI–96: Advances in Artificial Intelligence,* LNAI 1137. Springer, Dresden. pp. 67–70.

Fronhöfer, B. (1996e). Situational calculus, linear connection proofs and STRIPS-like planning: an experimental comparison. In P. Miglioli, U. Moscato, D. Mundici and M. Ornaghi (Eds.), *5-th Workshop on Theorem Proving with Analytic Tableaux and Related Methods,* LNAI 1071. Springer, Terrasini, Palermo pp. 193–209.

Gelfond, M., and V. Lifshitz (1993). Representing action and change by logic programs. *Journal of Logic Programming,* **17**, 301–321.

Girard, J.-Y. (1987). Linear Logic. *Theoretical Computer Science,* **50**, 1–102.

Girard, J.-Y. (1990). *Logic and Exception: A Few Remarks.*Technical report, Université Paris VII.

Green, C. (1967). Application of theorem proving to problem solving. In *IJCAI–1.* pp. 219–239.

Große, G., S. Hölldobler and J. Schneeberger (1992a). *Linear Deductive Planning.* Technical report AIDA–92–08, FG Intellektik, FB Informatik, TH Darmstadt.

Große, G., S. Hölldobler and J. Schneeberger (1996). Linear deductive planning. *Journal of Logic and Computation,* **6**(2), 233–262.

Große, G., S. Hölldobler, J. Schneeberger, U. Sigmund and M. Thielscher (1992b). Equational logic programming, actions, and change. In K. Apt (Ed.), *Proc. Joint Int. Conference and Symposium on Logic Programming JICSL'92.* MIT Press. pp. 177–191.

Herrmann, C.S, and M. Thielscher (1996a). *On Reasoning about Continuous Processes.*Technical Report AIDA–96–04, FG Intellektik, FB Informatik, TH Darmstadt.

Herrmann, C.S. and M. Thielscher (1996b). Reasoning about continuous processes. In B. Clancey and D. Weld (Eds.), *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI).* MIT Press, Portland. pp. 639–644.

Hölldobler, S. and M. Thielscher (1995). Computing change and specificity with equational logic programs. *Annals of Mathematics and Artificial Intelligence,* **14**, 99–133.

Hölldobler, S. (1996). Equational logic and theories of action. In P. Lucio, M. Martelli and M. Navarro (Eds.), *Proceedings of the APPIA–GULP–PRODE Joint Conference on Declarative Programming.* pp. 111–123.

Hölldobler, S, and J. Schneeberger (1990). A new deductive approach to planning. *New Generation Computing,* **8**, 225–244. A short version appeared in the Proceedings of the German Workshop on Artificial Intelligence, Informatik Fachberichte 216 pp. 63–73, 1989.

Hölldobler, S., and J. Schneeberger (1996). *Constraint Equational Logic Programming and Resource-based Partial Order Planning.* Technical Report WV-96-08, FG Wissensverarbeitung, Institut für KI, Fakultät Informatik, TU Dresden.

Hölldobler, S., and M. Thielscher (1993a). Actions and specificity. In D. Miller (Ed.), *Proceedings of the Int. Logic Programming Symposium (ILPS).* MIT Press, Vancouver. pp. 164–180.

Hölldobler, S. and M. Thielscher (1993b). On logic, change, and specficity. In B. Fronhöfer (Ed.), *Proceedings of the Workshop on Reasoning about Action & Change at the Int. Joint Conference on Artificial Intelligence.* pp. 3–7.

Hölldobler, S, and M. Thielscher (1996). *Properties vs. Resources: Solving Simple Frame Problems.* Technical Report AIDA-96-03, FG Intellektik, FB Informatik, TH Darmstadt.

Jacopin, E. (1993a). Classical AI planning as theorem proving: the case of a fragment of linear logic. In *AAAI Fall Syposium on "Automated Deduction in Non Classical Logics".* AAAI Press, Palo Alto pp. 62–66.

Jacopin, E. (1993b). *Construire des plans en utilisant le calcul des Séquents pour un fragment de la logique linéaire.*Technical Report, Laforia-IBP, Univ. P. et M. Curie, Paris.

Kowalski, R. (1979). *Logic For Problem Solving.* North Holland, New York.

Lifshitz, V. (1986). On the Semantics of STRIPS. In M. George and A. Lansky (Eds.), *Workshop on Reasoning About Actions and Plans.* Morgan Kaufmann. pp. 1–8.

Masseron, M. C. Tollu and J. Vauzeilles (1990). Generating plans in linear logic. In *Foundations of Software Technology and Theoretical Computer Science,* LNCS 472. Springer pp. 63–75.

Masseron, M, C. Tollu and J. Vauzeilles (1993). Generating plans in linear logic I: actions as proofs. *Theoretical Computer Science,* **113**, 249–370.

McCarthy, J., and P. Hayes (1969). Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer, D. Michie (Eds.), *Machine Intelligence,* Vol. 4. Edinburgh University Press. pp. 463–502.

Nilsson, N.J. (1982). *Principles of Artificial Intelligence.* Springer.

Penberthy, J., and D. Weld (1992). UCPOP: a sound, complete, partial order planner for ADL. In *KR-92.* pp. 103–114.

Reiter, R. (1991). The frame problem in the situation calculus. In V. Lifschitz (Ed.), *Artificial Intelligence and Mathematical Theory of Computation.* Academic Press. pp. 359–380.

Sandewall, E. (1994). *Features and Fluents. The Representation of Knowledge about Dynamical Systems,* Volume 30 of *Oxford Logic Guides.* Oxford University Press.

Schneeberger, J. (1992). *Plan Generation by Linear Deduction.* PhD thesis, Technische Hochschule Darmstadt, Fachbereich Informatik.

Schubert, L. (1990). Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In H. Kyberg, R. Loui and G. Carlson (Eds.), *Knowledge Representation and Defeasable Knowledge.* Kluwer Academic Press, Bosten.

Shapiro, S. (1992). *Encyclopedia of Artificial Intelligence.* J. Wiley & Sons.

Sigmund, U.C. (1992). *LLP – Lineare Logische Programmierung.* Technical Report AIDA-92-18, FG Intellektik, FB Informatik, TH Darmstadt.

Thielscher, M. (1992). *AC1-Unifikation in der Linearen Logischen Programmierung.* TASSO-Report 42, FG Intellektik, FB Informatik, TH Darmstadt (in German).

Thielscher, M. (1994a). An analysis of systematic approaches to reasoning about actions and change. In P. Jorrand, and V. Sgurev (Eds.), *Int. Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA).* World Scientific, Singapore pp. 195–204.

Thielscher, M. (1994b). Representing actions in equational logic programming. In P.V. Hentenryck (Ed.), *Proceedings of the Int. Conference on Logic Programming (ICLP).* MIT Press, Santa Margherita Ligture. pp. 207–224.

Thielscher, M. (1995). Computing ramifications by postprocessing. In C.S. Mellish (Eds.), *Proceedings of the Int. Joint Conference on Artificial Intelligence (IJCAI)* Morgan Kaufmann, Montreal. pp. 1994–2000.

Thielscher, M. (1996). Causality and the qualification problem. In L.C. Aiello and S.C. Shapiro (Eds.), *Proceedings of the Int. Conference on Principles of Knowledge Representation and Reasoning(KR)* Morgan Kaufmann, Cambridge.

Thielscher, M. (1997). Ramification and causality. *Artificial Intelligence Journal.* (To appear. A preliminary version is available as Technical Report TR–96–003, ICSI, Berkeley, CA).

Thielscher, M., and T. Schaub (1995). Default reasoning by deductive planning. *Journal of Automated Reasoning.* Special Issue on the Automation of Commonsense and Nonmonotonic Reasoning, **15**(1), 1–409.

Vauzeilles, J., and C. Fouqueré 1993. *Linear logic and taxonomic networks.* Technical report, Département de mathématiques et informatique, Avenue J.B. Clément, F–93430 Villetaneuse.

Received December 1996

B. Fronhöfer studied mathematics at the LMU München 1973– 1982 and received his Diploma in 1982. From January 1979 to October 1982 he was a part-time employee of SIEMENS AG, München, where he worked on natural language processing. In November 1982 B. Fronhöfer joined the Automated Reasoning Research Group at Institute of Informatics at TU-München. He obtained a Ph.D. from the Institut National Polytechnique de Grenoble in 1989 and his habilitation from the Technical University Munich in February 1995. His current research interests include automatic deduction, plan generation and logic programming.

PLANO GENERAVIMAS, NAUDOJANT TIESINIŲ SĄSAJŲ METODĄ

Bertranas FRONHIOFERIS

Straipsnyje nagrinėjamos plano generavimo, naudojant dedukcijos metodus, problemos. Detaliai aptariami logikos metodų taikymo šiam uždaviniui spręsti trūkumai ir siūloma, kaip tuos trūkumus galima apeiti, panaudojus tiesinių sąsajų metodą.