TOWARDS PLANNING AS CONCURRENCY

Ozan Kahramanoğulları
Computer Science Institute, University of Leipzig
International Center for Computational Logic, TU Dresden
email: ozan@informatik.uni-leipzig.de

ABSTRACT
We present a purely logical framework to planning where we bring the sequential and parallel composition in the plans to the same level, as in process algebras. The problem of expressing causality, which is very challenging for common logics and traditional deductive systems, is solved by resorting to a recently developed extension of multiplicative exponential linear logic with a self-dual, non-commutative operator. We present an encoding of the conjunctive planning problems in this logic, and provide a constructive soundness and completeness result. We argue that this work is the first, but crucial, step of a uniform deductive formalism that connects planning and concurrency inside a common language, and allows to transfer methods from concurrency to planning.

KEY WORDS
Planning, Concurrency, Proof theory

1 Introduction

Concurrency and planning are two fields of computer science that evolved independently, aiming at solving tasks that are similar in nature, but different in perspective: while planning formalisms focus on finding a plan (process), if there *exists* such a plan, that solves a given planning problem; the focus in process algebra is on queries with the same nature, but universally quantified, such as dead-lock freeness or verification of security protocols. This requires an arsenal of mathematical methods, e.g., bisimulation, which respects the parallel behavior of actions and allows for an analysis of equivalence of processes.

In a process algebra parallel and sequential composition are at the same level since they are equivalently important notions for expressing concurrent processes. However, in planning the emphasis is on sequential composition. Although, the parallel behavior[1] of actions have been studied in partial order planners, e.g., [22], and Graph-plan approach, e.g.,[2], aiming at performance improvement, these investigations remained distant from the underlying logical framework. In [18], Kautz *et. al.* presented a planner, called Blackbox, which combines features of Graph-plan and planning as logical satisfiability. This approach resulted in a better computational performance, but did not provide a logical analysis of plans. [11] is an overview of these approaches, and also others.

In this paper we establish the first, but crucial, step of our long term goal of providing a purely logical common language to planning and concurrency inside a uniform deductive formalism. This way it will be possible to import the techniques of concurrency to planning which will provide a structural analysis of plans. Similar ideas were previously brought by Pym *et. al.* in [23] where, abstracting away from the conditions and effects of actions, they propose a process algebra like method for reasoning about actions, and argue that their method can be used to compare plans. However, in a perspective that captures the conditions and effects of actions resources are crucial from a concurrency point of view. Linear logic, which is a resource conscious logic, is a natural candidate for this task: weakening and contraction are controlled, i.e., the multiplicative conjunction $\otimes$ is not idempotent ("$a \otimes a \vdash a$" is not provable), whereas conjunction $\wedge$ in classical logic is idempotent. Furthermore, linear logic is widely recognized as a logic of concurrency (see, e.g., [21]). However, although, parallel composition can be mapped to the commutative *par* operator of linear logic, sequential composition does not find a natural interpretation.

With respect to resource consciousness, the relation between logic, actions and causality has been studied by various authors: in [1], Bibel imposes a syntactical condition called linearity on proofs, which requires that each literal is engaged in at most one connection. In [12], based on multiset rewriting, Hölldobler and Schneeberger introduce an equational Horn logic where states are represented by an AC1 function symbol. In [20], Masseron *et al.* applies multiplicative fragment of Girard's linear logic [6] to resource conscious planning by axiomatizing the actions as proper axioms. Linear logic approach to planning is studied further by various authors [13, 19, 5]. In [7], it is shown to be equivalent to the approaches in [1] and [12].

In this paper, we further elaborate on the linear logic approach to planning, aiming at providing a common language for planning and concurrency. For this purpose we employ system NEL of *the calculus of structures* [9]. The calculus of structures is a proof theoretical formalism, which is a generalization of the one-sided sequent calculus

[1] Our intention by parallel behavior is different then the approaches to concurrent actions in the logical AI literature, e.g., in [24], where concurrency is defined over the parametrized time spans shared by the actions.

with the gain of interesting proof theoretical properties. It was conceived to represent the logical system BV, which is a conservative extension of multiplicative linear logic with a self-dual, non-commutative operator, called *seq*. System NEL [10] is an extension of BV with the exponentials of linear logic. In other words, system NEL is an extension of multiplicative exponential linear logic with the self-dual, non-commutative operator *seq*. These systems can not be designed in the sequent calculus, as was shown by Tiu [27].

In [3], Bruscoli showed that there is a correspondence between system BV and a fragment of the process algebra CCS: the sequential composition corresponds to the non-commutative operator *seq*. Parallel composition is naturally mapped to the commutative linear logic operator *par*. Every terminating computation in the process algebra corresponds to a proof, and for every *process expression* provable, there is a corresponding terminating computation. However, as it is the case in CCS, there only the actions (labels) are included in the language, but not the resources that are consumed and produced by the actions.

In the following, we present an encoding of the conjunctive (multiset rewriting) planning problems in the language of NEL, where plans are not extracted from the proof of a planning problem, but become explicit premises of derivations which are analogous to the process expressions of [3]: by exploiting the non-commutative operator *seq*, and the commutative logical operator *par* of system NEL, we are able to observe plans, where the parallel behavior of plans is respected. This allows system NEL to give the complete operational semantics of our method. This way we establish the first step of a uniform formalism that connects concurrency and planning, and make it possible to transfer methods from concurrency to planning.

The rest of the paper is organized as follows: we begin with recapitulating notions and notations of conjunctive planning problems and system NEL. We then present an encoding of the conjunctive planning problems in the language of NEL and show that our encoding is correct for plans that are sequences of actions. Following this, we extend our correctness result to plans which include parallel composition. We elaborate further on our approach and conclude with future work and discussions. Space restrictions do not permit us to give the complete proof of the theorems, we refer to [16].

2 Planning Problems

Following [7, 20], a *planning domain* is given by: (1) a set of constants representing atomic properties of the world which we call *fluents* and denote by small letters; (2) a set of *transition rules (actions)* [2] that are multiset[3] rewrite rules; (3) *states* which are multisets of fluents. A conjunctive planning problem $\mathscr{P}$ is then given by $\langle \mathcal{I}, \mathcal{G}, \mathscr{A}, \mathcal{F} \rangle$ where $\mathcal{I} : \{\!| r_1, \ldots, r_m |\!\}$ is a multiset of fluents called *initial state*. The multiset $\mathcal{G} : \{\!| g_1, \ldots, g_n |\!\}$ of fluents is the *goal state*. $\mathscr{A}$ is a finite set of *actions* of the form $a : \{\!| c_1, \ldots, c_p |\!\} \to \{\!| e_1, \ldots, e_q |\!\}$, where $\{\!| c_1, \ldots, c_p |\!\}$ and $\{\!| e_1, \ldots, e_q |\!\}$ are multisets of fluents called *conditions* and *effects*, respectively, and a is the *name* of the action. $\mathcal{F} = \{f_1, \ldots, f_h\}$ is the set of all the fluents that appear in $\mathcal{I}$, $\mathcal{G}$ and $\mathscr{A}$.

An action is applicable in a state $\mathcal{S}$ iff $\{\!| c_1, \ldots, c_p |\!\} \dot{\subseteq} \mathcal{S}$. The application of an action a to a state $\mathcal{S}$ is defined by the function Φ as follows.

$$\Phi(a, \mathcal{S}) = (\, \mathcal{S} \dot{-} \{\!| c_1, \ldots, c_p |\!\}\,) \,\dot{\cup}\, \{\!| e_1, \ldots, e_q |\!\}$$

A goal $\mathcal{G}$ is satisfied iff there is a *plan (structure)* p, i.e., a sequence of actions $p = \langle a_1; \ldots; a_k \rangle$, which transforms the initial state into a state $\mathcal{S} \dot{\supseteq} \mathcal{G}$ such that $\Phi(a_k, \ldots, \Phi(a_1, \mathcal{I}) \ldots) = \mathcal{S}$. If there exists such a plan p, then p is a solution for the planning problem $\mathscr{P}$. Then we say p *solves* $\mathscr{P}$. We denote the empty plan with $\circ$. If it is more convenient, $\Phi(a_k, \ldots, \Phi(a_1, \mathcal{I}) \ldots)$ will be abbreviated with $\Phi(p, \mathcal{I})$. The *length of a plan* is the number of actions in that plan.

Now, to illustrate the above theory on a planning problem, let us look at the following example which is a modification of an example from [7]. Suppose Bert is thirsty and wants to get some lemonade (l) from a vending machine. The lemonade costs 50 cents (f). Bert has a dollar bill (d) in his pocket. Because the vending machine accepts only 50 cents coins, Bert has to get change for his dollar. The problem of getting the lemonade can be described as a planning problem with the initial state $\mathcal{I} : \{\!| d |\!\}$, the actions $c_d : \{\!| d |\!\} \to \{\!| f, f |\!\}$ and $b_l : \{\!| f |\!\} \to \{\!| l |\!\}$ that allow him to change a dollar for two 50 cents coins, and to buy a lemonade, respectively. The goal state in which Bert got the lemonade is given by $\mathcal{G} : \{\!| l |\!\}$.

Clearly, the solution to the problem is the plan in which at first Bert changes the dollar and then buys the lemonade: applying this plan to the initial state yields, first, the state $\{\!| f, f |\!\}$, and then $\{\!| f, l |\!\}$. As the goal is contained in the last state, the planning problem is solved.

3 The Calculus of Structures and NEL

In this section, we present the calculus of structures [9] and system NEL [10] which is a conservative extension of multiplicative exponential linear logic with a non-commutative operator.

There are countably many *atoms*, denoted by $a, b, c, \ldots$ The *structures*[4] of the language NEL are denoted by P, Q, R,

[2] We consider only propositional actions.

[3] Multisets are denoted by the curly brackets "$\{\!|$" and "$|\!\}$". $\dot{\cup}$, $\dot{-}$ and $\dot{\subseteq}$ denote the multiset operations corresponding to the usual set operations $\cup$, $-$ and $\subseteq$, respectively.

[4] The notion of a structure is similar to the notion of a formula or a sequent of the sequent calculus. However, a structure denotes an equivalence class of structures. Fore a formal elaboration of this notion, we refer the reader to [9].

Associativity

$$[\vec{R},[\vec{T}]] = [\vec{R},\vec{T}]$$
$$(\vec{R},(\vec{T})) = (\vec{R},\vec{T})$$
$$\langle\vec{R};\langle\vec{T}\rangle\rangle = \langle\vec{R};\vec{T}\rangle$$

Commutativity

$$[\vec{R},\vec{T}] = [\vec{T},\vec{R}]$$
$$(\vec{R},\vec{T}) = (\vec{T},\vec{R})$$

Singleton

$$[R] = (R) = \langle R\rangle = R$$

Units

$$[\circ,R] = R$$
$$(\circ,R) = R$$
$$\langle\circ;R\rangle = R$$
$$\langle R;\circ\rangle = R$$

Exponentials

$$??R = ?R$$
$$!!R = !R$$
$$?\circ = \circ$$
$$!\circ = \circ$$

Negation

$$\bar{\circ} = \circ$$
$$\overline{[R,T]} = (\overline{R},\overline{T})$$
$$\overline{(R,T)} = [\overline{R},\overline{T}]$$
$$\overline{\langle R;T\rangle} = \langle\overline{R};\overline{T}\rangle$$
$$\overline{?R} = !\overline{R}$$
$$\overline{!R} = ?\overline{R}$$
$$\overline{\overline{R}} = R$$

Figure 1. The equational system underlying System NEL.

$S\ldots$ and are generated by

$$R ::= \quad a \quad | \quad \circ \quad | \quad !R \quad | \quad ?R \quad | \quad \bar{R} \quad |$$
$$[\underbrace{R,\ldots,R}_{>0}] \quad | \quad (\underbrace{R,\ldots,R}_{>0}) \quad | \quad \langle\underbrace{R;\ldots;R}_{>0}\rangle$$

where a stands for any atom and $\circ$, the *unit*, is not an atom. A structure $[R_1,\ldots,R_h]$ is a *par structure*, $(R_1,\ldots,R_h)$ is a *times structure*, $\langle R_1;\ldots;R_h\rangle$ is a *seq structure*, $!R$ is called an *of-course structure*, and $?R$ is called a *why-not structure*; $\bar{R}$ is the *negation* of the structure R. Structures are considered to be equivalent modulo the relation $=$, which is the smallest congruence relation induced by the equations shown in Figure 1. A *structure context*, denoted as in $S\{\ \}$, is a structures with a hole that does not appear in the scope of negation. The structure R is a *substructure* of $S\{R\}$ and $S\{\ \}$ is its *context*. Context braces are omitted if no ambiguity is possible.

In the calculus of structures, an *inference rule* is a scheme of the kind $\rho\,\dfrac{T}{R}$, where ρ is the *name* of the rule, T is its *premise* and R is its *conclusion*. A typical (deep) inference rule has the shape $\rho\,\dfrac{S\{T\}}{S\{R\}}$ and specifies a step of rewriting by the implication $T \Rightarrow R$ inside a generic context $S\{\ \}$, which is linear implication[5] in our case. An inference rule is called an *axiom* if its premise is empty. Rules with empty contexts correspond to the case of the sequent calculus.

A (formal) *system* $\mathscr{S}$ is a set of inference rules. A derivation Δ in a certain formal system is a finite chain of instances of inference rules in the system. A derivation can consist of just one structure. The topmost structure in a derivation, if present, is called the *premise* of the derivation, and the bottommost structure is called its *conclusion*.

[5] Due to duality between $T \Rightarrow R$ and $\bar{R} \Rightarrow \bar{T}$, rules come in pairs of dual rules: a down-version and an up-version. For instance, the dual of the $\mathrm{ai}{\downarrow}$ rule is the cut rule. In this paper, we only consider the down rules, since the up rules, including the cut rule, are admissible.

$$\mathrm{ai}{\downarrow}\,\frac{S\{\circ\}}{S[a,\bar{a}]} \qquad \mathrm{s}\,\frac{S([R,T],U)}{S[(R,U),T]} \qquad \mathrm{q}{\downarrow}\,\frac{S\langle[R,T];[U,V]\rangle}{S[\langle R;U\rangle,\langle T;V\rangle]}$$

$$\circ{\downarrow}\,\frac{}{\circ} \qquad \mathrm{p}{\downarrow}\,\frac{S\{![R,T]\}}{S[!R,?T]} \qquad w{\downarrow}\,\frac{S\{\circ\}}{S\{?R\}} \qquad \mathrm{b}{\downarrow}\,\frac{S[?R,R]}{S\{?R\}}$$

Figure 2. System NEL

A derivation Δ whose premise is T, conclusion is R, and inference rules are in $\mathscr{S}$ will be written as $\Delta\dfrac{\|T}{R}\mathscr{S}$. Similarly, $\Pi\dfrac{\|}{R}\mathscr{S}$ will denote a *proof* Π which is a finite derivation whose topmost inference rule is an axiom.

The system in Figure 2 is called *Non-commutative Exponential Linear logic*, or system NEL. The rules of the system are *unit* ($\circ{\downarrow}$), *atomic interaction* ($\mathrm{ai}{\downarrow}$), *switch* (s), *seq* ($\mathrm{q}{\downarrow}$), *promotion* ($\mathrm{p}{\downarrow}$), *weakening* ($w{\downarrow}$), and *absorption* ($\mathrm{b}{\downarrow}$).

For system NEL, the cut rule has the shape $\mathrm{i}{\uparrow}\,\dfrac{S(R,\bar{R})}{S\{\circ\}}$.

Theorem 1 (Cut Elimination). [10] The rule $\mathrm{i}{\uparrow}$ is admissible for system NEL, in other words, for every proof $\Pi\dfrac{\|}{R}\mathsf{NEL}\cup\{\mathrm{i}{\uparrow}\}$, there is a proof $\Pi'\dfrac{\|}{R}\mathsf{NEL}$.

Theorem 2 (Decomposition). [26] For every derivation Δ in system NEL, there is a derivation Δ' where, seen bottom-up, first system $\{\mathrm{b}{\downarrow}\}$, then $\{w{\downarrow}\}$, and then $\{\mathrm{p}{\downarrow},\mathrm{s},\mathrm{q}{\downarrow},\mathrm{ai}{\downarrow}\}$ are applied.

There is a straightforward correspondence between structures not involving seq and formulae of multiplicative exponential linear logic (MELL). For example $![(?a,b),\bar{c},!\bar{d}]$ corresponds to $!\,((?a\otimes b)\parr c^{\perp}\parr !\,d^{\perp})$, and vice versa. Units 1 and $\perp$ are mapped into $\circ$, since $1 \equiv \perp$, when the rules mix and mix0 are added to MELL. For a proof of the above results, a more detailed discussion on the proof theory of NEL and the precise relation between NEL and MELL, the reader is referred to [26].

4 Planning with NEL

In this section, we present our encoding of the planning problems in the language of NEL and show that it is correct with respect to conjunctive planning problems.

Definition 3. The *sequential action structure* for an action, $a : \{\!|\, c_1,\ldots,c_p \,|\!\} \rightarrow \{\!|\, e_1,\ldots,e_q \,|\!\}$, denoted by $\mathcal{Q}$, is the structure $\langle[\bar{c}_1,\ldots,\bar{c}_p];a;[e_1,\ldots,e_q]\rangle$.

Definition 4. The *simple problem structure* P^s for an initial state $\mathcal{I} = \{\!| r_1, \ldots, r_m |\!\}$ and a goal state $\mathcal{G} = \{\!| g_1, \ldots, g_n |\!\}$ is the structure $[r_1, \ldots, r_m, \bar{g}_1, \ldots, \bar{g}_n]$.

Because an action can be executed arbitrarily many times, we employ the exponential "?" which retains a controlled contraction and weakening on the action structures. This way, we can duplicate an action structure by applying the $\mathrm{b}{\downarrow}$ rule, or annihilate it by applying the $w{\downarrow}$ rule during the search for the plans. This also allows us to make the interaction between the planning problems and actions explicit by prefixing a planning problem structure with "!": by applying the $\mathrm{p}{\downarrow}$ rule in proof search we allow an action structure to get inside and interact with a problem structure.

We can now define a planning problem in the language of NEL.

Definition 5. Let $\mathscr{P} = \langle \mathcal{I}, \mathcal{G}, \mathscr{A}, \mathcal{F} \rangle$ be a planning problem. The *sequential conjunctive planning problem structure* (shortly scpps) for $\mathscr{P}$, denoted by $\mathcal{P}^s$, is the structure

$$[?\mathcal{Q}_1, \ldots, ?\mathcal{Q}_k, !P^s, ?\bar{f}_1, \ldots, ?\bar{f}_h]$$

where $\mathcal{Q}_i$ $(1 \leq i \leq k)$ are the sequential action structures for the actions in $\mathscr{A}$, P^s is the simple problem structure for $\mathcal{I}$ and $\mathcal{G}$, and $\mathcal{F} = \{f_1, \ldots, f_h\}$.

Let us reconsider the conjunctive planning problem from Section 2. This planning problem can be expressed as the following scpps.

$$[\,?\langle \bar{d}; c_d; [f, f]\rangle\,,\ ?\langle \bar{f}; b_l; l\rangle\,,\ !\,[d, \bar{l}],\ ?d, ?f, ?l\,] \qquad (1)$$

The structures $?\langle \bar{d}; c_d; [f, f]\rangle$ and $?\langle \bar{f}; b_l; l\rangle$ are the sequential action structures for the actions c_d and b_l, respectively. The structure $[d, \bar{l}]$ is the simple problem structure for the initial state $\mathcal{I} = \{\!| d |\!\}$ and the goal state $\mathcal{G} = \{\!| l |\!\}$. The structures $?d, ?l$ and $?f$ correspond to $\mathcal{F} = \{d, f, l\}$.

In the following, we will show that searching for a certain kind of derivations where the conclusion is the scpps for a planning problem is equivalent to finding a solution for this planning problem. With the following lemmata, we will formally express the operational semantics of reaching a goal state and applying an action to a state in the language of NEL, respectively.

Lemma 6. *[16]* The following rule, called *termination*, is derivable in NEL.

$$tr\ \frac{S\{!\,P\}}{S[!\,\langle P; [r_1, \ldots, r_n, g_1, \ldots, g_s, \bar{r}_1, \ldots, \bar{r}_n]\rangle, ?\bar{f}_1, \ldots, ?\bar{f}_h]}$$

where $g_i \in \{f_1, \ldots, f_h\}$ for $1 \leq i \leq s$.

Lemma 7. *[16]* The following rule, called *action*, is derivable in NEL.

$$act\ \frac{S[\,?\langle[\bar{c}_1, \ldots, \bar{c}_p]; a; E\rangle\,,\ !\,\langle P; a; [R, E]\rangle\,]}{S[\,?\langle[\bar{c}_1, \ldots, \bar{c}_p]; a; E\rangle\,,\ !\,\langle P; [c_1, \ldots, c_p, R]\rangle\,]}$$

By employing the rules $action$ and $termination$ bottom-up, we can search for plans while going up in a derivation: the rule $action$ is applied till the multiset of negative atoms in the of-course structure denoting the simple problem structure is a submultiset of the multiset of positive atoms, where the rule $termination$ can be applied. After annihilating the of-course structures for the sequential action structures and excessive resources with the rule $w{\downarrow}$, such a derivation will then give a plan structure at the premise which is a solution for the planning problem. The following theorem proves that our encoding is correct.

Theorem 8. Let $\mathscr{P} = \langle \mathcal{I}, \mathcal{G}, \mathscr{A}, \mathcal{F} \rangle$ be a conjunctive planning problem and $\mathcal{P}^s$ the scpps for $\mathscr{P}$.

$$\text{There is a derivation } \begin{array}{c} !\,p \\ \|_{\mathsf{NEL}} \\ \mathcal{P}^s \end{array} \text{ iff the plan } p \text{ solves } \mathscr{P}.$$

Sketch of Proof: *[16]* If part of the proof is by induction on the length of the plan at the premise of the derivation. The base case follows from Lemma 6. The inductive case follows from Lemma 7 and Theorem 2. Only if part is with induction on the length of the plan. Take the derivation that was constructed at the if part. □

To illustrate the above ideas, let us return to our running example. Observe that the conclusion of the below derivation is the scpps in (1).

$$\begin{array}{l} tr \\ act \\ act \\ = \end{array} \frac{\begin{array}{l} w{\downarrow}^4\ \dfrac{!\,\langle c_d\,; b_l\rangle}{[\,?\langle \bar{d}; c_d; [f, f]\rangle\,,\ ?\langle \bar{f}; b_l; l\rangle\,,\ !\,\langle c_d\,; b_l\rangle,\ ?d, ?l\,]} \\ \dfrac{[\,?\langle \bar{d}; c_d; [f, f]\rangle\,,\ ?\langle \bar{f}; b_l; l\rangle\,,\ !\,\langle c_d\,; b_l\,; [f, l, \bar{l}]\rangle,\ ?d, ?f, ?l\,]}{[\,?\langle \bar{d}; c_d; [f, f]\rangle\,,\ ?\langle \bar{f}; b_l; l\rangle\,,\ !\,\langle c_d\,; [f, f, \bar{l}]\rangle,\ ?d, ?f, ?l\,]} \end{array}}{[\,?\langle \bar{d}; c_d; [f, f]\rangle\,,\ ?\langle \bar{f}; b_l; l\rangle\,,\ !\,\langle \circ\,; [d, \bar{l}]\rangle,\ ?d, ?f, ?l\,]}$$

The plan structure at the premise is a solution of our planning problem.

5 Parallel Plans

As well as sequential composition due to non-commutative *seq* operator, the language of NEL allows to express parallel composition of plans and actions by employing the commutative *par* operator. In this section, we further extend the notion of plans to the notion of parallel plans, and show that our encoding of the planning problems allows to capture parallel behavior in plans.

Definition 9. A *parallel plan structure* is a structure generated by

$$P ::= \circ \mid a \mid \langle\, P\,; P\,\rangle \mid [\,P\,, P\,]$$

where a denotes atoms representing actions.

Proposition 10. For every planning problem $\mathscr{P}$ given with $\mathcal{I}$, $\mathcal{G}$, and $\mathscr{A}$, and a plan $\langle a_1; \ldots; a_k \rangle$ that solves it, for some $s \leq k$, there is a planning problem $\mathscr{P}'$ given with $\mathcal{I}' = \Phi(a_s, \ldots, \Phi(a_1, \mathcal{I}) \ldots)$, $\mathcal{G}$ and $\mathscr{A}$ that is solved by $\langle a_{s+1}; \ldots; a_k \rangle$.

Sketch of Proof: Follows immediately from the definitions in Section 2. □

Proposition 11. Let $\mathcal{I}, \mathcal{S}_1, \mathcal{S}_2$ be states and p be a plan. $\Phi(p, \mathcal{I}) = \mathcal{S}_1$ iff $\Phi(p\,, \mathcal{I} \dot\cup \mathcal{S}_2) = \mathcal{S}_1 \dot\cup \mathcal{S}_2$.

Sketch of Proof: With induction on the length of p. □

Lemma 12. Let $\mathcal{I}_1 = \{\!|\, r_1, \ldots, r_m \,|\!\}$, $\mathcal{I}_2 = \{\!|\, r'_1, \ldots, r'_n \,|\!\}$, $\mathcal{S}_1 = \{\!|\, g_1, \ldots, g_p \,|\!\}$ and $\mathcal{S}_2 = \{\!|\, g'_1, \ldots, g'_q \,|\!\}$ be states and $p_1 = \langle a_1, \ldots, a_k \rangle$, $p_2 = \langle a'_1, \ldots, a'_{k'} \rangle$ be plans. Furthermore, let $\mathcal{Q}_1, \ldots, \mathcal{Q}_k, \mathcal{Q}'_1, \ldots, \mathcal{Q}'_{k'}$ be the sequential action structures for the actions $a_1, \ldots, a_k, a'_1, \ldots, a'_{k'}$. The following are equivalent.

(i) $\Phi(\, p_1, \Phi(\, p_2, \mathcal{I}_1 \dot\cup \mathcal{I}_2\,)) = \Phi(\, p_2, \Phi(\, p_1, \mathcal{I}_1 \dot\cup \mathcal{I}_2\,)) = \mathcal{S}$.

(ii) $\Phi(p_1, \mathcal{I}_1) = \mathcal{S}_1$ and $\Phi(p_2, \mathcal{I}_2) = \mathcal{S}_2$ such that $\mathcal{S} = \mathcal{S}_1 \dot\cup \mathcal{S}_2$.

(iii)
$$\begin{array}{c} \langle [p_1, p_2]; [g_1, \ldots, g_p, g'_1, \ldots, g'_q] \rangle \\ \| \{\mathtt{a1}\downarrow, \mathtt{q}\downarrow\} \\ [\mathcal{Q}_1, \ldots, \mathcal{Q}_k, \mathcal{Q}'_1, \ldots, \mathcal{Q}'_{k'}, r_1, \ldots, r_m, r'_1, \ldots, r'_n] \end{array}$$

Sketch of Proof:
$(i) \Rightarrow (ii)$: Let $\Phi(p_1, \mathcal{I}_1) = \mathcal{S}'$. From Proposition 11, we have
$\Phi(p_2\,, \Phi(p_1\,, \mathcal{I}_1 \dot\cup \mathcal{I}_2)) = \Phi(p_2\,, \mathcal{S}' \dot\cup \mathcal{I}_2) = \mathcal{S}$.
Assume that (i) holds and (ii) does not hold.
This can only be the case when there are fluents in $\mathcal{S}'$ that are not present in $\mathcal{I}_2$ and consumed by p_2, but this contradicts with $\Phi(\, p_2, \Phi(\, p_1, \mathcal{I}_1 \dot\cup \mathcal{I}_2\,)) = \mathcal{S}$

$(ii) \Rightarrow (iii)$: Observe that (ii) implies that there are the following derivations.

$$\begin{array}{c} \langle p_1; [g_1, \ldots, g_p] \rangle \\ \Delta_1 \| \{\mathtt{a1}\downarrow, \mathtt{q}\downarrow\} \\ [\mathcal{Q}_1, \ldots, \mathcal{Q}_k, r_1, \ldots, r_m] \end{array} \qquad \begin{array}{c} \langle p_2; [g'_1, \ldots, g'_q] \rangle \\ \Delta_2 \| \{\mathtt{a1}\downarrow, \mathtt{q}\downarrow\} \\ [\mathcal{Q}'_1, \ldots, \mathcal{Q}'_{k'}, r'_1, \ldots, r'_n] \end{array}$$

Take the following derivation.

$$\begin{array}{c} \mathtt{q}\downarrow \dfrac{\langle [p_1, p_2]; [g_1, \ldots, g_p, g'_1, \ldots, g'_q] \rangle}{[\langle p_1; [g_1, \ldots, g_p] \rangle, \langle p_2; [g'_1, \ldots, g'_q] \rangle]} \\ \Delta_1 \Big\| \\ [\mathcal{Q}_1, \ldots, \mathcal{Q}_k, r_1, \ldots, r_m, \langle p_2; [g'_1, \ldots, g'_q] \rangle] \\ \Delta_2 \Big\| \\ [\mathcal{Q}_1, \ldots, \mathcal{Q}_k, \mathcal{Q}'_1, \ldots, \mathcal{Q}'_{k'}, r_1, \ldots, r_m, r'_1, \ldots, r'_n] \end{array}$$

$(iii) \Rightarrow (i)$: The following derivations together with Theorem 8 prove the result.

$$\begin{array}{l} = \dfrac{\langle p_1; p_2 \rangle}{\langle [p_1, \circ]; [\circ, p_2] \rangle} \\ \mathtt{q}\downarrow \dfrac{}{[\langle p_1; \circ \rangle, \langle \circ; p_2 \rangle]} \\ = \dfrac{}{[p_1, p_2]} \end{array} \qquad \begin{array}{l} = \dfrac{\langle p_2; p_1 \rangle}{\langle [\circ, p_2]; [p_1, \circ] \rangle} \\ \mathtt{q}\downarrow \dfrac{}{[\langle \circ; p_1 \rangle, \langle p_2; \circ \rangle]} \\ = \dfrac{}{[p_1, p_2]} \end{array}$$

□

Definition 13. A parallel plan structure P *solves a planning problem* $\mathscr{P}$, if, for all the derivations

$$\begin{array}{c} p \\ \| \{\mathtt{q}\downarrow\} \\ P \end{array}$$

where p is a plan structure, p solves $\mathscr{P}$.

To illustrate these ideas let us return to our running example. However, this time Bert is not only thirsty but also hungry. Since he is equipped with the action that allows him to *get a candy-bar* (c) for 50 cents from the vending machine, this should not be a problem. Then, once he has a lemonade and a candy bar, he can *have lunch* which makes him *happy* (h). Consider the following scpps

$$[\,?\langle \bar{d}; c_d; [f, f] \rangle\,,\ ?\langle \bar{f}; b_l; l \rangle\,,\ ?\langle \bar{f}; g_c; c \rangle\,,\ ?\langle [\bar{l}, \bar{c}]; h_l; h \rangle\,,\ !\,[d, \bar{h}]\,,\ ?d, ?f, ?l, ?c, ?h\,]$$

where $?\langle \bar{f}; g_c; c \rangle$ and $?\langle [\bar{l}, \bar{c}]; h_l; h \rangle$, respectively, are the sequential action structures for the actions *get a candy-bar* and *have lunch*, respectively. It is easy to observe that the parallel plan structure

$$\langle c_d\,; [b_l\,, g_c]; h_l \rangle$$

solves the above planning problem. The following theorem formally justifies that there is a derivation which provides this parallel plan structure at the premise.

Theorem 14. Let $\mathscr{P}$ be a planning problem and $\mathcal{P}^s$ be the scpps for $\mathscr{P}$. If P is a parallel plan structure that solves a planning problem $\mathscr{P}$, then there is a derivation of the following form.

$$\begin{array}{c} !\,P \\ \Delta \| \mathsf{NEL} \\ \mathcal{P}^s \end{array}$$

Sketch of Proof: Let $p = \langle a_1, \ldots, a_k \rangle$ be a plan structure such that there is a derivation $\begin{array}{c} p \\ \| \{\mathtt{q}\downarrow\} \\ P \end{array}$. From Theorem 8 there is a derivation $\begin{array}{c} !\,\langle a_1, \ldots, a_k \rangle \\ \| \mathsf{NEL} \\ \mathcal{P}^s \end{array}$ and from Theorem 2, there is a derivation of the following form

$$\begin{array}{c} !\,\langle a_1, \ldots, a_k \rangle \\ \Delta_1 \| \{\mathtt{a1}\downarrow, \mathtt{q}\downarrow\} \\ !\,[\mathcal{Q}_1, \ldots, \mathcal{Q}_k, r_1, \ldots, r_m, \bar{g}_1, \ldots, \bar{g}_n] \\ \Delta_2 \| \{\mathtt{p}\downarrow, \mathtt{w}\downarrow, \mathtt{b}\downarrow\} \\ \mathcal{P}^s \end{array}$$

where $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ are the sequential action structures for the actions $a_1, \ldots, a_k$. It remains to prove that there is a derivation

$$\begin{array}{c} P \\ \Delta_3 \, [\!|\{\mathtt{ai}\downarrow, \mathtt{q}\downarrow\} \\ [\mathcal{Q}_1, \ldots, \mathcal{Q}_k, r_1, \ldots, r_m, \bar{g}_1, \ldots, \bar{g}_n] \end{array}$$

We will construct the derivation Δ_3 with structural induction on P.

– If $P = \circ$ or $P = a$, then take Δ_1.

– If $P = \langle P_1; P_2 \rangle$, then there must be a plan $p = \langle p_1; p_2 \rangle$ that solves $\mathscr{P}$ such that $\begin{array}{c} p_1 \\ [\!|\{\mathtt{q}\downarrow\} \\ P_1 \end{array}$ and $\begin{array}{c} p_2 \\ [\!|\{\mathtt{q}\downarrow\} \\ P_2 \end{array}$ where $p_1 = \langle a_1, \ldots, a_{k'} \rangle$ and $p_2 = \langle a_{k'+1}, \ldots, a_k \rangle$. Proposition 10 and Theorem 8 give the derivation

$$\begin{array}{c} [\langle P_1; [t_1, \ldots, t_h] \rangle, \langle [\bar{t}_1, \ldots, \bar{t}_h]; P_2 \rangle] \\ [\!|\{\mathtt{ai}\downarrow, \mathtt{q}\downarrow\} \\ [\mathcal{Q}_1, \ldots, \mathcal{Q}_k, r_1, \ldots, r_m, \bar{g}_1, \ldots, \bar{g}_n] \end{array} .$$

– If $P = [P_1, P_2]$ then there must be plans $p = \langle p_1; p_2 \rangle$ and $p' = \langle p_2; p_1 \rangle$ that solve $\mathscr{P}$ such that $\begin{array}{c} p_1 \\ [\!|\{\mathtt{q}\downarrow\} \\ P_1 \end{array}$ and $\begin{array}{c} p_2 \\ [\!|\{\mathtt{q}\downarrow\} \\ P_2 \end{array}$. Lemma 12 and Theorem 8 give the derivation

$$\begin{array}{c} [P_1, P_2] \\ [\!|\{\mathtt{ai}\downarrow, \mathtt{q}\downarrow\} \\ [\mathcal{Q}_1, \ldots, \mathcal{Q}_k, r_1, \ldots, r_m, \bar{g}_1, \ldots, \bar{g}_n] \end{array} .$$

□

Corollary 15. Let $\mathscr{P}$ be a planning problem and $\mathcal{P}^s$ be the scpps for $\mathscr{P}$. If P is a parallel plan structure that solves a planning problem $\mathscr{P}$, then the structure $[\mathcal{P}^s, ?\bar{P}]$ has a proof in NEL.

Sketch of Proof: Result follows immediately from Theorem 14. □

6 Discussion

We presented an encoding of the conjunctive planning problems in the language of NEL where plans are not extracted from the proof of a derivation, but they are explicit premises of derivations. Furthermore, we showed that our encoding is expressive enough to capture plans where the parallel behavior of actions can be captured.

A direct consequence of this work is the establishment of a bridge between concurrency theory and planning, which makes it possible to employ methods form concurrency in the lines of [3].

Another connection of our work with concurrency is via Labeled Event Structures (LES). LES is a model for concurrency [25] which has been studied for a class of linear logic proofs in [8]. In LES the causality between events is expressed as a partial order and the non-determinism is obtained by a conflict relation. In LES one focuses on events and their relations, abstracting away from states. This way they provide a behavioral model of concurrency. In [15] we presented a framework where conjunctive planning problems are expressed as structures in multiplicative exponential linear logic in the calculus of structures, and provided an algorithm to extract partial order plans that exhibit LES semantics from the proofs of these structures. This way we established an explicit correspondence between partial order plans and proofs.

The parallel plans that we derived in this paper can be observed as *N-free* partial orders[6]. However, in the partial order plans that we extract from linear logic proofs are not restricted to *N-free* partial orders. One of our future objective is to associate to every planning problem a LES which represents the independence and causality of all actions performable in all different derivations produced by the search for a proof of a planning problem. This is a concurrent model of the possible computations. Future work includes a general comparison of these methods.

We have implemented the proof search for the systems in the calculus of structures, and also a planner which implements the above ideas in the lines of [14, 17]. These implementations, mainly in system Maude [4], are available[7] for download. However, at the moment they are plausible only for planning problems of toy size.

Acknowledgments This work has been supported by the DFG Graduiertenkolleg 446. Lutz Straßburger provided me a previous version of the encoding of the planning problems in BV. I would like to thank Alessio Guglielmi, Steffen Hölldobler, and Lutz Straßburger.

References

[1] Wolfgang Bibel. A deductive solution for plan generation. In *New Generation Computing*, pages 115–132. 1986.

[2] A. Blum and Furst M. Fast planning through planning graph analysis". In *Artificial Intelligence*, volume 90, pages 281–300. 1997.

[3] Paola Bruscoli. A purely logical account of sequentiality in proof search. In Peter J. Stuckey, editor, *Logic Programming, 18th International Conference*, volume 2401 of *Lecture Notes in Computer Science*, pages 302–316. Springer-Verlag, 2002.

[6] A partial order R is *N-free* iff $\{(a,b),(c,d),(c,b)\} \subseteq R$ implies $(a,d) \in R$.

[7] http://www.informatik.uni-leipzig.de/ ~ozan/maude_cos.html

[4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. The Maude 2.0 system. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications, Proceedings of the 14th International Conference,*, volume 2706. Springer, 2003.

[5] S. Cresswell, A. Smaill, and J. Richardson. Deductive synthesis of recursive plans in linear logic. In *ECP*, 1999.

[6] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

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

[8] Alessio Guglielmi. *Abstract Logic Programming in Linear Logic Independence and Causality in a First Order Calculus*. PhD thesis, Universita di Pisa – Genova, 1996.

[9] Alessio Guglielmi. A system of interaction and structure. Technical Report WV-02-10, TU Dresden, 2002. to appear in ACM Transactions on Computational Logic.

[10] Alessio Guglielmi and Lutz Straßburger. A non-commutative extension of MELL. In M. Baaz and A. Voronkov, editors, *LPAR 2002*, volume 2514 of *Lecture Notes in Artificial Intelligence*, pages 231–246. Springer-Verlag, 2002.

[11] J. Hoffmann. *Utilizing Problem Structure in Planning: A Local Search Approach*, volume 2854 of *LNAI*. Springer Verlag, 2003.

[12] S. Hölldobler and J. Schneeberger. A new deductive approach to planning. In *New Generation Computing*, pages 225–244. 1990.

[13] Éric Jacopin. Classical AI planning as theorem proving: The case of a fragment of linear logic. In *AAAI Fall Symposium on Automated Deduction in Nonstandard Logics*, pages 62–66, Palo Alto, California, 1993. AAAI Press Publications.

[14] Ozan Kahramanoğulları. Implementing system BV of the calculus of structures in maude. In Laura Alonso i Alemany and Paul Égré, editors, *Proceedings of the ESSLLI-2004 Student Session*, pages 117–127, Université Henri Poincaré, Nancy, France, 2004.

[15] Ozan Kahramanoğulları. A new logical notion of partial order planning. Technical report, TU Dresden, 2004.

[16] Ozan Kahramanoğulları. Plans as formulae with a non-commutative operator. Technical report, TU Dresden, 2004.

[17] Ozan Kahramanoğulları. System BV without the equalities for unit. In *Proceedings of the 19th International Symposium on Computer and Information Sciences, ISCIS'04*, Lecture Notes in Computer Science. Springer, 2004. to appear.

[18] Henry Kautz and Bart Selman. Pushing the envelope: Planning, propositional logic and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 1194–1201, Menlo Park, 1996. AAAI Press / MIT Press.

[19] N. Kobayashi and A. Yonezawa. Reasoning on actions and change in linear logic programming. Technical report, Department of Information Science, University of Tokyo, 1993.

[20] M. Masseron, C. Tollu, and J. Vauzeilles. Generating plans in linear logic. In *Foundations of Software Technology and Theoretical Computer Science*, volume 472 of *Lecture Notes in Computer Science*, pages 63–75. Springer-Verlag, 1990.

[21] Dale Miller. The π-calculus as a theory in linear logic: Preliminary results. In E. Lamma and P. Mello, editors, *Proceedings of the 1992 Workshop on Extensions to Logic Programming*, number 660 in LNCS, pages 242–265. Springer-Verlag, 1993.

[22] M. E. Pollack, D. Joslin, and M. Paolucci. Flaw selection strategies for partial-order planning. In *Journal of Artificial Intelligence Research*, volume 6, pages 223–262. 1997.

[23] David Pym, Louise Pryor, and David Murphy. A note on processes for plan-execution and powerdomains for plan-comparison. Technical report, Department of Computer Science, Queen Mary and Westfield College, University of London, 1996.

[24] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 2–13. Cambridge, Morgan Kaufmann, 1996.

[25] Vladimiro Sassone, Morgens Nielsen, and Glynn Winskel. Models for concurrency: Towards a classification. In *Theoretical Computer Science*, volume 170 (1–2), pages 297–348. 1996.

[26] Lutz Straßburger. *Linear Logic and Noncommutativity in the Calculus of Structures*. PhD thesis, TU Dresden, 2003.

[27] Alwen Fernanto Tiu. Properties of a logical system in the calculus of structures. Technical Report WV-01-06, Technische Universität Dresden, 2001.