Abstract. System NEL is a conservative extension of multiplicative ex-
ponential linear logic with a self-dual, non-commutative operator. In this paper, we express plans as logical formulae by using this sequential op-
erator. We present an encoding of the conjunctive planning problems in
the language of NEL where plans are not extracted from the proofs, but
become explicit premises of derivations. We then extend the notion of a
plan to cover parallel composition by employing a commutative logical
operator. This way, like in concurrency, sequential and parallel composi-
tion come to the same level where the reasoning is done in a purely logical
way. We provide a constructive correctness proof and an implementation
of system NEL in Maude, and argue that this work is the primary step
for providing a common language for planning and concurrency.

1 Introduction

Concurrency and planning are two fields of computer science that evolved in-
dependently, aiming at solving tasks that are similar in nature, but different in
perspective: while planning formalisms focus on finding a plan (process) that
solves a planning problem, the focus in process algebras, such as CCS [17], is
concepts like dead-lock freeness and verification of security protocols. In such a
perspective, a notion of equivalence of processes, e.g., bisimilarity, which respects
the parallel behavior of the processes gains importance.

In a process algebra parallel and sequential composition are at the same
level, since they are equivalently important notions for expressing concurrent
processes. On the other hand, in planning the emphasis in the literature is on
the sequential composition. In the case of the partial order planners, the plan
is computed by heuristic methods, in a way distant from the underlying logical
framework, and then the computed plan is linearized to be executed. However,
parallel composition is natural in logic due to the commutative logical operators.
In this paper, in a resource conscious setting, we establish the first, but crucial,
step for our long term goal of a common language for concurrency and planning,
inside a uniform deductive formalism.

With respect to resource consciousness, the relation between logic, actions
and causality has been studied by various authors: In [1], Bibel imposes a syntac-
tical condition called linearity on proofs, which requires that each literal is en-
gaged 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 [16], 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, 15, 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 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. Tiu showed in [21] that this system can not be designed in the sequent calculus.

System NEL [10] is an extension of system BV with the exponentials of linear logic. In [2], Bruscoli showed that there is a correspondence between system BV and a fragment of CCS: the sequential composition corresponds to the non-commutative operator seq. Parallel composition is naturally mapped to the commutative linear logic operator par. 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. This way, similar to [2], by exploiting the non-commutative operator of system NEL, and the commutative logical operator par, we are able to observe concurrent plans, where the parallelism between plans is respected. Since our encoding is propositional, no unification mechanism is needed. This allows system NEL to give the complete operational semantics of our method, and establish the first step of a uniform formalism that connects concurrency and planning. This way, it becomes possible to transfer methods from concurrency to planning.

In [14], we presented an implementation of the system BV in Maude 2 [3]. By extending those modules by accommodating exponentials, in this paper, we also present an implementation of system NEL, where search for proofs and derivations becomes possible.

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. After presenting a sketch of a Maude 2 implementation of system NEL, we conclude with a brief discussion.
2 Planning Problems

Following [7, 16], 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) \(^1\) that are multiset\(^2\) rewrite rules; (3) states which are multisets of fluents. A conjunctive planning problem \(P\) is then given by \(I, G, A, F\) where \(I: \{ r_1, \ldots, r_m \}\) is a multiset of fluents called initial state. The multiset \(G: \{ g_1, \ldots, g_n \}\) of fluents is the goal state. \(A\) is a finite set of actions of the form \(a : \{ c_1, \ldots, c_p \} \rightarrow \{ e_1, \ldots, e_q \}\), 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. \(F = \{ f_1, \ldots, f_h \}\) is the set of all the fluents that appear in \(I, G\) and \(A\).

An action is applicable in a state \(S\) iff \(\{ c_1, \ldots, c_p \} \subseteq S\). The application of an action \(a\) to a state \(S\) is defined by the function \(\Phi\) as follows.

\[
\Phi(a, S) = (S \setminus \{ c_1, \ldots, c_p \}) \cup \{ e_1, \ldots, e_q \}.
\]

A goal \(G\) 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 \(S\), i.e., \(\Phi(a_k, \ldots, \Phi(a_1, I) \ldots) = S\) such that \(G \subseteq S\). If there exists such a plan \(p\), then \(p\) is a solution for the planning problem \(P\). Then we say \(p\) solves \(P\). We denote the empty plan with \(\emptyset\). If it is more convenient, \(\Phi(a_k, \ldots, \Phi(a_1, I) \ldots)\) will be abbreviated with \(\Phi(p, I)\). 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 \(I: \{ d \}\), the actions \(c_d : \{ d \} \rightarrow \{ f, f \}\) and \(b_l : \{ f \} \rightarrow \{ l \}\) 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 \(G: \{ l \}\).

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.

\(^1\) We consider only propositional actions.

\(^2\) Multisets are denoted by the curly brackets \("\{\}\)". \(\setminus\), \(-\) and \(\subseteq\) denote the multiset operations corresponding to the usual set operations \(\cup\), \(-\) and \(\subseteq\), respectively.
### Assosciativity

\[
\begin{align*}
[[R,T],U] &= [R,[T,U]] \\
(R,T),U &= (R,(T,U)) \\
(R;\langle T; U \rangle) &= \langle R;\circ \rangle = \langle R;\circ \rangle = \langle R;\circ \rangle = \langle R;\circ \rangle = \langle R;\circ \rangle
\end{align*}
\]

### Units

\[
\begin{align*}
[\circ, R] &= R \\
(R, T) &= (\overline{R}, \overline{T}) \\
(R; T) &= (\overline{R}; \overline{T})
\end{align*}
\]

### Negation

\[
\begin{align*}
\overline{\circ} &= \circ \\
\overline{R} &= R
\end{align*}
\]

### Commutativity

\[
\begin{align*}
[R,T] &= [T,R] \\
(R,T) &= (T,R) \\
(R; \circ) &= (\circ; R) = R
\end{align*}
\]

### Exponentials

\[
\begin{align*}
??R &= ??R \\
!!R &= !R
\end{align*}
\]

### Singleton

\[
\begin{align*}
[R] &= (R) = \langle R \rangle = R \\
?!R &= ?R
\end{align*}
\]

### Contextual Closure

\[
\begin{align*}
?[R] &= [T,R] \\
(R,T) &= (T,R) \\
(R;\circ) &= (\circ; R)
\end{align*}
\]

---

### Fig. 1. The equational system underlying System NEL.

3 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. For a formal elaboration of this notion, we refer the reader to [9].
the implication $T \Rightarrow R$ inside a generic context $S\{ \}$, which is linear implication\(^4\) 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* $\mathcal{S}$ is a set of inference rules. A derivation $\Delta$ 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*. A derivation $\Delta$ whose premise is $T$, conclusion is $R$, and inference rules are in $\mathcal{S}$ will be written as $\Delta \vdash_{\mathcal{S}} T \Rightarrow R$. Similarly, $\Rightarrow \vdash_{\mathcal{S}} R$ will denote a *proof* $\Pi$ 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* ($\text{ai}_\downarrow$), *switch* ($\text{s}$), *seq* ($\text{q}_\downarrow$), *promotion* ($\text{p}_\downarrow$), *weakening* ($\text{w}_\downarrow$), and *absorption* ($\text{b}_\downarrow$).

For system NEL, the cut rule has the shape $i \downarrow S(R, \overline{R}) \vdash_{\mathcal{S}} S\{\}$.\(^5\)

**Theorem 1 (Cut Elimination).** [10] The rule $i \downarrow$ is admissible for system NEL, in other words, for every proof $\Pi \vdash_{\text{NEL} \cup \{i\}} R$, there is a proof $\vdash_{\text{NEL}} R$.

**Theorem 2 (Decomposition).** [19] For every derivation $\Delta$ in system NEL, there is a derivation $\Delta'$ where, seen bottom-up, first system $\{\text{b}_\downarrow\}$, then $\{\text{w}_\downarrow\}$, and then $\{\text{p}_\downarrow, \text{s}, \text{q}_\downarrow, \text{ai}_\downarrow\}$ are applied.

There is a straightforward correspondence between structures not involving seq and formulae of multiplicative exponential linear logic (MELL). For example $![(?a, b), ?c, !d]$ corresponds to $!((?a \otimes b) \otimes c) \otimes !d)$, and vice versa. Units 1 and

\(^4\) Due to duality between $T \Rightarrow R$ and $\overline{R} \Rightarrow \overline{T}$, rules come in pairs of dual rules: a down-version and an up-version. For instance, the dual of the $\text{ai}_\downarrow$ rule is the cut rule. In this paper, we only consider the down rules, since the up rules, including the cut rule, are admissible.
In this section, we present our encoding of the planning problems in the language of \( \text{NEL} \) and show that it is correct with respect to conjunctive planning problems.

**Definition 1.** The sequential action structure for an action, \( a : \{ c_1, \ldots, c_p \} \rightarrow \{ e_1, \ldots, e_q \} \), denoted by \( Q \), is the structure \( \{ [c_1, \ldots, c_p] : a; [e_1, \ldots, e_q] \} \).

**Definition 2.** The simple problem structure \( P^s \) for an initial state \( I = \{ r_1, \ldots, r_m \} \) and a goal state \( G = \{ g_1, \ldots, g_n \} \) is the structure \( \{ r_1, \ldots, r_m, g_1, \ldots, g_n \} \).

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 \( b\| \) rule, or annihilate it by applying the \( w\| \) 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 \( p\| \) 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 \( \text{NEL} \).

**Definition 3.** Let \( \mathcal{P} = \{ I, G, \mathcal{A}, F \} \) be a planning problem. The sequential conjunctive planning problem structure (shortly scpps) for \( \mathcal{P} \), denoted by \( P^s \), is the structure

\[
?Q_1, \ldots, ?Q_k, !P^s, ?\bar{f}_1, \ldots, ?\bar{f}_h
\]

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

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

\[
[ ?\langle d; c_d; [f, f] \rangle, ?\langle \bar{f}; b_l; l \rangle, !\{ d, \bar{l} \}, ?d, ?f, ?l ]
\]  

The structures \( ?\langle d; c_d; [f, f] \rangle \) 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 \( I = \{ d \} \) and the goal state \( G = \{ l \} \). The structures \( ?d, ?l \) and \( ?f \) correspond to \( F = \{ d, f, l \} \).

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 \( \text{NEL} \), respectively.
Lemma 1. The following rule is derivable in NEL.

\[
S[! P] \quad \text{termination}
\]

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

Proof: Take the following derivation.

\[
\begin{align*}
\text{ai} \vdash S[! P] \\
\vdots \\
\text{ai} \vdash ![\langle P; [g_1, \ldots, g_s, \bar{g}_1, \ldots, \bar{g}_s] \rangle] \\
\text{q} \vdash ![\langle P; [g_1, \ldots, g_s] \rangle, \bar{g}_1, \ldots, \bar{g}_s] \\
\text{p} \vdash ![\langle P; [g_1, \ldots, g_s] \rangle, \bar{g}_1] \\
\vdots \\
\text{at} \vdash ![\langle P; r_1, \ldots, r_n, g_1, \ldots, g_s, \bar{r}_1, \ldots, \bar{r}_n \rangle], f_1, \ldots, f_h]
\end{align*}
\]

\[\square\]

Lemma 2. The following rule is derivable in NEL.

\[
S[? \langle [c_1, \ldots, c_p]; a; E \rangle, ![P; [a; E, R]] \rangle] \\
S[? \langle [c_1, \ldots, c_p]; a; E \rangle, ![P; [c_1, \ldots, c_p, R]] \rangle]
\]

Proof: Take the following derivation.

\[
\begin{align*}
\text{q} \vdash S[? \langle [c_1, \ldots, c_p]; a; E \rangle, ![P; [a; E, R]] \rangle] \\
\text{ai} \vdash S[? \langle [c_1, \ldots, c_p]; a; E \rangle, ![\langle [a; E, R] \rangle] \langle P; [c_1, \ldots, c_p, R] \rangle \rangle] \\
\vdots \\
\text{q} \vdash S[? \langle [c_1, \ldots, c_p]; a; E \rangle, ![\langle [a; E, R] \rangle] \langle P; [c_1, \ldots, c_p, R] \rangle \rangle] \\
\text{q} \vdash S[? \langle [c_1, \ldots, c_p]; a; E \rangle, ![\langle [a; E, R] \rangle] \langle P; [c_1, \ldots, c_p, R] \rangle \rangle] \\
\text{p} \vdash S[? \langle [c_1, \ldots, c_p]; a; E \rangle, ![\langle [a; E, R] \rangle] \langle P; [c_1, \ldots, c_p, R] \rangle \rangle] \\
\text{b} \vdash S[? \langle [c_1, \ldots, c_p]; a; E \rangle, ![\langle [a; E, R] \rangle] \langle P; [c_1, \ldots, c_p, R] \rangle \rangle]
\end{align*}
\]
By employing the rules \textit{action} and \textit{termination} bottom-up, we can search for plans while going up in a derivation: the rule \textit{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 \textit{termination} can be applied. After annihilating the of-course structures for the sequential action structures and excessive resources with the rule \textit{w}\textsubscript{1}, 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.

\textbf{Theorem 3.} Let \(\mathcal{P} = \langle \mathcal{I}, \mathcal{G}, \mathcal{A}, \mathcal{F} \rangle\) be a conjunctive planning problem and \(\mathcal{P}^s\) the scpps for \(\mathcal{P}\). There is a derivation

\[
!p \\
\text{NEL} \\
\mathcal{P}^s
\]

iff the plan \(p\) solves \(\mathcal{P}\).

\textbf{Proof:} (\(\Rightarrow\)) Proof with induction on the length of the plan at the premise of the derivation. If there is a derivation with the empty plan \(\circ\) at the premise of the derivation, then it follows from Lemma 1 that there must be a derivation of the following form.

\[
\text{termination} \quad \frac{\text{w}\downarrow \circ}{\text{w}\downarrow [?Q_1, \ldots, ?Q_k, !\langle o; r_1, \ldots, r_n, f_1, \ldots, f_k \rangle], ?f_1, \ldots, ?f_h]}
\]

It follows that for \(\mathcal{I} \supseteq \{r_1, \ldots, r_n\} = \mathcal{G}\), plan \(\circ\) with length 0 solves the planning problem. Turning to the induction step, we assume that if there is a derivation

\[
\langle a_1; \ldots; a_j \rangle \\
\text{NEL} \\
[?Q_1, \ldots, ?Q_k, !\langle r_1, \ldots, r_m, g_1, \ldots, g_n \rangle, ?f_1, \ldots, ?f_h]
\]

then the plan \(\langle a_1; \ldots; a_j \rangle\) solves the planning problem \(\langle \mathcal{I}, \mathcal{G}, \mathcal{A}, \mathcal{F} \rangle\) where \(\mathcal{A} = \{a_1, \ldots, a_k\}, \mathcal{I} = \{r_1, \ldots, r_m\}\) and \(\mathcal{G} = \{g_1, \ldots, g_n\}\). If there is such a derivation \(\Delta\), then it follows from Theorem 2 that there is a derivation of the following form.
Consider the planning problem given with \( \mathcal{A}, \mathcal{T}' = \{ c_1, \ldots, c_p, r_{q+1}, \ldots, r_m \} \) \( \mathcal{G} \) and \( \mathcal{F}' \) such that \( \mathcal{F} \subseteq \{ f'_1, \ldots, f'_{h'} \} = \mathcal{F}' \). Suppose there is a derivation with a plan of length \( j + 1 \) at the premise. Then, from Lemma 2 there must exist a derivation of the following form.

\[
\begin{align*}
&\langle a; a_1, \ldots, a_j \rangle \\
&\Delta_1 \parallel\{a; a_1, a_j\} \\
&\langle a; [Q_1, \ldots, Q_j, r_1, \ldots, r_m, g_1, \ldots, g_n, \bar{f}_1, \ldots, \bar{f}_{h''}] \rangle \\
&\Delta_2 \parallel\{p_1\} \\
&\langle [Q_1, \ldots, Q_j, r_1, \ldots, r_m, g_1, \ldots, g_n], \bar{f}_1, \ldots, \bar{f}_{h''} \rangle \\
&\Delta_3 \parallel\{w_1\} \\
&\langle [Q_1, \ldots, Q_j, r_1, \ldots, r_m, g_1, \ldots, g_n], \bar{f}_1, \ldots, \bar{f}_{h} \rangle \\
&\Delta_4 \parallel\{b_1\} \\
&\langle [Q_1, \ldots, Q_j, r_1, \ldots, r_m, g_1, \ldots, g_n], \bar{f}_1, \ldots, \bar{f}_{h} \rangle \\
\end{align*}
\]

Observe that there is an action \( a : \{ c_1, \ldots, c_p \} \rightarrow \{ r_1, \ldots, r_q \} \). Since

\( \mathcal{T}' = \{ c_1, c_p \} \cup \{ r_1, \ldots, r_q \} = \{ r_1, \ldots, r_m \} \),

it follows that \( \langle a; a_1, \ldots, a_k \rangle \) with length \( k + 1 \) is a solution for the planning problem.
Proof with induction on the length of the plan $p$. If $p$ is the empty plan, then it must be the case that $\mathcal{I} = \{ r_1, \ldots, r_m \} \supseteq \mathcal{G}$. Take the derivation (2). Turning to the induction step we assume that the result holds for a plan with $k$ number of actions. Suppose there is a plan $\langle a; a_1; \ldots; a_k \rangle$ that solves the planning problem given with $\mathcal{I} = \{ r_1, \ldots, r_m, c_1, \ldots, c_p \}$ and $\mathcal{G} = \{ g_1, \ldots, g_n \}$. Hence we find an action

$$a : \{ c_1, \ldots, c_p \} \rightarrow \{ r_1, \ldots, r_q \}, \text{ where } q < m,$$

and a planning problem given with

$$\mathcal{I}' = \{ r_1, \ldots, r_q, r_{q+1}, \ldots, r_m \}, \quad \mathcal{G} = \{ g_1, \ldots, g_n \}$$

which is solved by $\langle a_1; \ldots; a_k \rangle$. Since we have the derivation $\Delta$ in (3) from the induction hypothesis, the derivation (4) proves the result. \hfill $\Box$

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).

\[
\begin{align*}
& \text{termination} \quad w \downarrow^4 \quad \! \langle c_d; b_l \rangle \\
& \text{action} \quad [ ?(d; c_d; [f; f]), \quad ?(f; b_l; l), \quad !\langle c_d; b_l \rangle, \quad ?d, ?l ] \\
& \text{action} \quad [ ?(d; c_d; [f; f]), \quad ?(f; b_l; l), \quad !\langle c_d; b_l; [f; f, l] \rangle, \quad ?d, ?f, ?l ] \\
& \quad = [ ?(d; c_d; [f; f]), \quad ?(f; b_l; l), \quad !\langle c_d; b_l\rangle, \quad ?d, ?f, ?l ]
\end{align*}
\]

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

5 Concurrent Plans

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

**Definition 4.** A concurrent plan structure is a structure generated by

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

where $o$ denotes atoms representing actions.

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

Proof: Follows immediately from the definitions in Section 2. \qed

**Proposition 2.** Let \( \mathcal{I}, S_1, S_2 \) be states and \( p \) be a plan.
\( \Phi(p, \mathcal{I}) = S_1 \) iff \( \Phi(p, \mathcal{I} \cup S_2) = S_1 \cup S_2 \).

Proof: With induction on the length of \( p \). \qed

**Lemma 3.** Let \( \mathcal{I}_1 = \{r_1, \ldots, r_m\} \), \( \mathcal{I}_2 = \{r'_1, \ldots, r'_n\} \), \( S_1 = \{g_1, \ldots, g_p\} \) and \( S_2 = \{g'_1, \ldots, g'_q\} \) 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 \( Q_1, \ldots, Q_k, Q'_1, \ldots, Q'_k \) be the sequential action structures for the actions \( a_1, \ldots, a_k, a'_1, \ldots, a'_k \). The following are equivalent.

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

(ii) \( \Phi(p_1, \mathcal{I}_1) = S_1 \) and \( \Phi(p_2, \mathcal{I}_2) = S_2 \) such that \( S = S_1 \cup S_2 \).

(iii) \[
\langle [p_1, p_2]; [g_1, \ldots, g_p, g'_1, \ldots, g'_q] \rangle
\|
\langle \{a_1, a_1\} \rangle
\mid
\langle Q_1, \ldots, Q_k, Q'_1, \ldots, Q'_k, r_1, \ldots, r_m, r'_1, \ldots, r'_n \rangle
\]

Proof:

(i) \( \Rightarrow \) (ii): Let \( \Phi(p_1, \mathcal{I}_1) = S' \). From Proposition 2, we have
\( \Phi(p_2, \Phi(p_1, \mathcal{I}_1 \cup \mathcal{I}_2)) = \Phi(p_2, S' \cup \mathcal{I}_2) = S \).

Assume that (i) holds and (ii) does not hold. This can only be the case when there are fluents in \( S' \) that are not present in \( \mathcal{I}_2 \) and consumed by \( p_2 \), but this contradicts with \( \Phi(p_2, \Phi(p_1, \mathcal{I}_1 \cup \mathcal{I}_2)) = S \).

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

\[
\langle p_1; [g_1, \ldots, g_p] \rangle \quad \Delta_1 \| \langle \{a_1, a_1\} \rangle
\mid
\langle Q_1, \ldots, Q_k, r_1, \ldots, r_m \rangle
\]

\[
\langle p_2; [g'_1, \ldots, g'_q] \rangle \quad \Delta_2 \| \langle \{a_1, a_1\} \rangle
\mid
\langle Q'_1, \ldots, Q'_k, r'_1, \ldots, r'_n \rangle
\]

Take the following derivation.

\[
q \mid \langle [p_1, p_2]; [g_1, \ldots, g_p, g'_1, \ldots, g'_q] \rangle
\]

\[
\langle [p_1; [g_1, \ldots, g_p]], (p_2; [g'_1, \ldots, g'_q]) \rangle
\]

\[
\Delta_1 \| \langle Q_1, \ldots, Q_k, r_1, \ldots, r_m, (p_2; [g'_1, \ldots, g'_q]) \rangle
\]

\[
\Delta_2 \| \langle Q_1, \ldots, Q_k, Q'_1, \ldots, Q'_k, r_1, \ldots, r_m, r'_1, \ldots, r'_n \rangle
\]
(iii) ⇒ (i): The following derivations together with Theorem 3 prove the result.

\[ q \downarrow \frac{\langle p_1; p_2 \rangle}{\langle [p_1; o]; [o; p_2] \rangle} \]

\[ q \downarrow \frac{\langle [p_1; o]; [o; p_2] \rangle}{[p_1; p_2]} \]

\[ q \downarrow \frac{\langle p_2; p_1 \rangle}{\langle [o; p_2]; [p_1; o] \rangle} \]

\[ q \downarrow \frac{\langle [o; p_1]; [p_2; o] \rangle}{[p_1; p_2]} \]

\[ \square \]

**Definition 5.** A concurrent plan structure \( P \) solves a planning problem \( \mathcal{P} \), if, for all the derivations

\[ p \frac{\{a_1\}}{P} \]

where \( p \) is a plan structure, \( p \) solves \( \mathcal{P} \).

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

\[ [?\langle d; c; d, f; f \rangle, ?\langle f; b; l \rangle, ?\langle f; g; c \rangle, ?\langle l, e; h \rangle, !\langle d, h \rangle, ?f, ?l, ?c, ?h] \]

where \( ?\langle f; g; c \rangle \) and \( ?\langle l, e; h \rangle \), respectively, are the sequential action structures for the actions get a candy-bar and have lunch, respectively. It is easy to observe that the concurrent plan structure

\[ \langle c; d, b, g; c; h \rangle \]

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

**Theorem 4.** Let \( \mathcal{P} \) be a planning problem and \( \mathcal{P}^* \) be the scpps for \( \mathcal{P} \). If \( P \) is a concurrent plan structure that solves a planning problem \( \mathcal{P} \), then there is a derivation of the following form.

\[ !P \]

\[ \frac{\Delta_{\text{NEL}}}{\mathcal{P}^*} \]

**Proof:** Let \( p = \langle a_1, \ldots, a_k \rangle \) be a plan structure such that there is a derivation \( p \frac{\{a_1\}}{P} \). From Theorem 3 there is a derivation \( !\langle a_1, \ldots, a_k \rangle \frac{\text{NEL}}{\mathcal{P}^*} \) and from Theorem
2, there is a derivation of the following form

\[ \begin{align*}
\phi(a_1, \ldots, a_k) \\
\Delta_1 \models \{a_1, q_1\} \\
\phi(Q_1, \ldots, Q_k, r_1, \ldots, r_m, g_1, \ldots, g_n) \\
\Delta_2 \models \{p_1, w_1, b_1\} \\
\end{align*} \]

where \(Q_1, \ldots, Q_k\) are the sequential action structures for the actions \(a_1, \ldots, a_k\).

It remains to prove that there is a derivation

\[ P \models \{a_1, q_1\} \]
\[ [Q_1, \ldots, Q_k, r_1, \ldots, r_m, g_1, \ldots, g_n] \]

We will construct the derivation \(\Delta_3\) with structural induction on \(P\).

- If \(P = o\) or \(P = a\), then take \(\Delta_1\).

- If \(P = \langle P_1; P_2 \rangle\), then there must be a plan \(p = \langle p_1; p_2 \rangle\) that solves \(\varphi\) such that

\[\begin{align*}
p_1 & \models \{q_1\} \\
p_2 & \models \{q_1\} \\
P_1 & \models \{q_1\} \\
P_2 & \models \{q_1\} \\
\end{align*}\]

where \(p_1 = \langle a_1, \ldots, a_{k'} \rangle\) and \(p_2 = \langle a_{k'+1}, \ldots, a_k \rangle\).

Let \(Q_1, \ldots, Q_k\) be the sequential action structures for the actions \(a_1, \ldots, a_k\). Then from Proposition 1 and Theorem 3, there must be a derivation of the following form.

\[ a_{i1} \downarrow \langle p_1; p_2 \rangle \]
\[ a_{i1} \downarrow \langle p_1; [t_1, \ldots, t_h, \bar{t}_1, \ldots, \bar{t}_h]; p_2 \rangle \]
\[ q_{i1} \downarrow \langle p_1; [t_1, \ldots, t_h, \langle \langle \bar{t}_1, \ldots, \bar{t}_h \rangle; p_2 \rangle \rangle \]
\[ q_i \downarrow \langle p_1; [t_1, \ldots, t_h]; \langle \langle \bar{t}_1, \ldots, \bar{t}_h \rangle; p_2 \rangle \rangle \]
\[ \models \{a_{i1}, q_1\} \]
\[ [Q_1, \ldots, Q_{k'}, r_1, \ldots, r_{m'}, g_1, \ldots, g_{n'}, \langle \bar{t}_1, \ldots, \bar{t}_h \rangle; p_2] \]
\[ \models \{a_{i1}, q_1\} \]
\[ [Q_1, \ldots, Q_{k'}, r_1, \ldots, r_{m'}, g_1, \ldots, g_{n'}, Q_{k'+1}, \ldots, Q_k, r_{m'+1}, \ldots, r_m, g_{n'+1}, \ldots, g_n] \]
\[ [Q_1, \ldots, Q_k, r_1, \ldots, r_m, g_1, \ldots, g_n] \]

It follows that the induction hypothesis gives the derivations

\[ \begin{align*}
\Delta_4 \models \{a_{i1}, q_1\} \quad \text{and} \quad \Delta_5 \models \{a_{i1}, q_1\} \\
\end{align*}\]
\[ \begin{align*}
[Q_1, \ldots, Q_{k'}, r_1, \ldots, r_{m'}, g_1, \ldots, g_{n'}] \\
[Q_{k'+1}, \ldots, Q_k, r_{m'+1}, \ldots, r_m, g_{n'+1}, \ldots, g_n] \\
\end{align*}\]
Take the derivation
\[
\begin{align*}
\text{ai} \vdash (P_1; P_2) \\
\vdash \ldots \\
\text{q} \vdash (P_1; [t_1, \ldots, t_h, t_{\ell_1}, \ldots, t_{g_1}]); P_2) \\
\vdash [\{P_1; [t_1, \ldots, t_h], [t_{\ell_1}, \ldots, t_{g_1}]; P_2\}] \\
\{\Delta_1, \Delta_2\} \{\text{ai}, \text{q}\} \\
[Q_1, \ldots, Q_k, r_1, \ldots, r_m, g_1, \ldots, g_n]
\end{align*}
\]

- If $P = [P_1, P_2]$ then there must be plans $p = (p_1; p_2)$ and $p' = (p_2; p_1)$ that solve $\mathcal{P}$ such that
  \[
  \begin{align*}
  &p_1 \parallel \{q_1\} \quad \text{and} \quad p_2 \parallel \{q_1\} \\
  &P_1 \qquad \quad P_2
  \end{align*}
  \]

From Lemma 3 and Theorem 3 there must be a derivation of the following form.
\[
[p_1, p_2] \\
\parallel \{\text{ai}, \text{q}\} \\
[Q_1, \ldots, Q_k', r_1, \ldots, r_m', g_1, \ldots, g_n', P_2] \\
\parallel \{\text{ai}, \text{q}\} \\
= [Q_1, \ldots, Q_k', r_1, \ldots, r_m', g_1, \ldots, g_n', Q_{k'+1}, \ldots, Q_k, r_{m'+1}, \ldots, r_m, g_{n'+1}, \ldots, g_n]
\]

It follows that induction hypothesis gives the derivations
\[
\begin{align*}
P_1 \Delta_1 \parallel \{\text{ai}, \text{q}\} \quad \text{and} \\
\Delta_2 \parallel \{\text{ai}, \text{q}\} \\
[Q_1, \ldots, Q_{k'}, r_1, \ldots, r_{m'}, g_1, \ldots, g_{n'}] \\
[Q_{k'+1}, \ldots, Q_k, r_{m'+1}, \ldots, r_m, g_{n'+1}, \ldots, g_n]
\end{align*}
\]

Take the derivation
\[
\begin{align*}
P_1 \Delta_1 \parallel \{\text{ai}, \text{q}\} \\
\Delta_2 \parallel \{\text{ai}, \text{q}\} \\
[Q_1, \ldots, Q_k, r_1, \ldots, r_m, g_1, \ldots, g_n]
\end{align*}
\]

\[\square\]

**Corollary 1.** Let $\mathcal{P}$ be a planning problem and $\mathcal{P}^*$ be the scpps for $\mathcal{P}$. If $P$ is a concurrent plan structure that solves a planning problem $\mathcal{P}$, then the structure $[\mathcal{P}^*, ?P]$ has a proof in NEL.

**Proof:** Result follows immediately from Theorem 4.
6 Logical Strategies

We did not make any use of the rule $s$ in our derivations. However, this rule can be employed to logically enforce strategies for backward or forward search for plans, where the search starts from the initial state or the goal state. For instance, consider the following schemes of derivations with the sequential action structures and simple problem structures at the premise.

\[
\begin{align*}
    s &\quad [(c_1, \ldots, c_p); a; [e_1, \ldots, e_q]] \\
    \vdots &\quad \vdots \\
    s &\quad [(c_1, \ldots, c_p); a; [e_1, \ldots, e_q]]
\end{align*}
\]

\[
\begin{align*}
    s &\quad [r_1, \ldots, r_m, (g_1, \ldots, g_n)] \\
    \vdots &\quad \vdots \\
    s &\quad [r_1, \ldots, r_m, g_1, \ldots, g_n]
\end{align*}
\]

By applying these schemes to all the sequential action structures and to the simple planning problem structure in a scpps the search for plans is forced to be forward search since only the atoms in a par structure can interact with the negative atoms in the condition of an action. In order to observe this, the reader can try to apply the induction argument of the proof of Theorem 3 backward by starting from the goal state and applying the last action: this does not work due to possible atoms in the goal state that are not necessarily produced by the last action of the plan.

The same method can be analogously applied for enforcing a backward search strategy.

7 Implementation in Maude

Besides other properties, one interesting and important property which distinguishes calculus of structures from the sequent calculus is deep inference: the calculus of structures does not depend on a notion of main connective, and like in term rewriting, inference rules are deeply applicable inside expressions. This makes it possible to express logical systems as term rewriting systems modulo equality [11, 14].

The language Maude [3] allows implementing term rewriting systems modulo equational theories due to the built in very fast matching algorithm that supports different combinations of associative, commutative equational theories, also with the presence of units. Another important feature of Maude that makes it a plausible platform for implementing systems of the calculus of structures is the availability of the search function since the 2.0 release of Maude. This function implements breadth-first search which is vital for complete search for derivations and proofs.

Exploiting these features, we implemented system NEL in Maude 2 within two modules: the first module converts an arbitrary NEL structure into a structure in negation normal form, i.e., to a structure where there are no occurrences of units and where negation is pushed to the atoms. The second module implements the inference rules of the system modulo associativity, commutativity
and equalities for unit(s). Different modules for the systems in the calculus of structures including NEL are available for download at http://www.informatik.uni-leipzig.de/~ozan/maude_cos.html/.

8 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 explicit premises of derivations. Furthermore, we showed that our encoding is expressive enough to capture concurrent plans where plans respecting parallelism can also be obtained.

A direct consequence of this work is the establishment of a bridge between concurrency theory and planning, which allows to apply methods from concurrency to planning: labeled event structures is a model for concurrency [18] which has been studied for linear logic proofs in the sequent calculus proofs in [8]. Future work includes investigating a labeled event structure semantics in our derivations which will result in a notion of plan equivalence.

Our results show that NEL can serve as an operational semantics for conjunctive planning problems. In conjunction with our implementation of system NEL, our result is, with future development, also of practical interest. We conjecture that the rules action and termination can be expressed in Maude as conditional rewrite rules due to the matching of atoms with opposite polarities. However, such an implementation will disregard the concurrent plans. Because system NEL is undecidable [20] an implementation for practical purposes that captures concurrent plans will require introduction of strategies at the Maude meta-level [4].

References


