Let's plan it deductively!

W. Bibel
Technical University Darmstadt, Germany

Abstract

The paper describes a transition logic, TL, and a deductive formalism for it. It shows how various important aspects (such as ramification, qualification, specificity, simultaneity, indeterminism etc.) involved in planning can be modelled in TL in a rather natural way. (The deductive formalism for) TL extends the linear connection method proposed earlier by the author by embedding the latter into classical logic, so that classical and resource-sensitiv reasoning coexist within TL. The attraction of a logical and deductive approach to planning is emphasised and the state of automated deduction briefly described.

1 Introduction

Artificial Intelligence (AI, or intellectics [Bibel, 1992a]) aims at creating artificial intelligence. Were there no natural intelligence, the sentence would be meaningless to us. Hence understanding natural intelligence by necessity has always been among the goals of intellectics (and is also the goal of cognitive science).

Different points of view for approaching the goal of creating artificial intelligence have been distinguished [Kushmerick, 1996]. Logicism [Nilsson, 199l], cognitivism [Laird *et* a/., 1987], and situated action [Agre, 1995] are three out of several such points of view. In a nutshell, the logicistic point of view argues that man can describe his creations (including an artificial intelligence) only by natural linguistic, hence logical means; thus any way towards artificial intelligence must in some sense be a logical one. This author is strongly committed to the logicistic approach. As a consequence he believes that any other approach is in fact a logicistic one in disguise.

Intelligence has many features. Clearly one of them is the ability to plan ahead in time. Intuitively, planning is logical reasoning of some kind. All the more one might expect that planning is the domain where logic and its deductive machinery excel. The fact is that it does not. There are many software systems in everyday use solving planning tasks, but to the author's best knowledge none of them is based on logic and has a deductive component. Does this imply that logic is irrelevant for planning and for artificial intelligence for that matter?

While intelligence implies the ability for planning, the converse has not necessarily to be true. It very much depends on what kind of planning is meant. In a fixed and relatively restricted domain (such as text layout) planning may well be realized in a purely functional way and with standard programming techniques. But functional (or procedural) programming has its limits as we enter more complex and unpredictable domains; in particular it will never be able to produce a behavior which rightly deserves to be named "intelligent" (surely as a user of computers you noticed the stupidity of text layout systems). Section 7, as well as numerous texts in the literature, give arguments for this statement. It also gives reasons which explain the resistance of the software industry to a bolder move into a logic technology for planning and for other applications. In other words, logic is essential for intelligent planning in the true sense of the term, but industry is not ready to build intelligent systems.

It is not the task of intellecticians to lament about this state of affairs but rather to prepare for the coming day when the market will be ripe for a broader use of a truly intelligent technology and to develop the best possible technological basis for it. In fact, if we are frank there is yet a lot to be developed before we can comfortably go out to industry and offer a coherent set of methods for dealing with the many facets of intelligence including planning.

In the present paper I review the state of the art in deductive planning with an emphasis on the contributions from research groups influenced by my own work. While much of the work in deductive planning has focused on representational issues we have always approached the problem with the necessary and available deductive techniques in mind. Since the methods and systems growing out of our work have finally achieved a leading position in the deduction community by winning the CADE-96 competition in automated theorem proving with the SETHEO system [Lets *et* a/., 1992; Moser *et* a/., 1997], we are perhaps also well placed to import the best possible techniques into the planning

community. In other words, the paper will focus on deductive planning as well as on the underlying deduction techniques. Since the author sees planning as just one among a number of aspects for achieving artificial intelligence, the case for deductive planning is presented in this paper in form of a paradigm case for achieving the grander goal of artificial intelligence. The paper will therefore not only point the way to intelligent planning but to some extent also the author's proposed way to artificial intelligence (the "it" in the title).

In the next section we introduce the logical language used in our approach and discuss the deductive aspects thereafter. The resulting computational logic is called *transition logic* (TL) which has classical as well as resource-sensitive features. Section 4 shows what TL has to do with planning and computation (or with temporal prediction or postdiction for that matter). Section 5 compares the logic with other known logics. Section 6 shows how the various aspects involved in reasoning about actions and causality can be taken into account within TL. Specifically, we discuss ramification, qualification, specificity, simultaneity, in determinism, continuity, hierarchies etc. Finally, we briefly describe the tensions between the specialistic and logistic approaches in AI and explain it by outlining the underlying pattern. Given the impressive recent achievements in automated deduction we conclude with making a case for a logical path towards an artificial intelligence.

2 A logical language

Any textbook on AI also contains some introduction to first-order logic so that we may assume the reader to be familiar with it. Only to communicate our notational conventions we mention that there are objects named by constants *(a,b,c),* (n-ary) functions named by function symbols *(f,g,h)* and (n-ary) relations named by predicate symbols (P, Q, *R).* Terms (r, s,t), built from variables (x,y, z, ranging over objects), constants and function symbols, again denote objects. Literals *(K, L)* are relations among objects or the negation $(\neg)$ thereof. They correspond to simple factual sentences in natural language (such as "John is married to the mother of BUT).

For building more complicated sentences represented as formulas (F, G, *H)* we use the well-known classical (logical) operators $\wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$ as well as the resource-sensitive operators & (non-idempotent conjunction), | (non-idempotent exclusive disjunction), and $\Rightarrow$ (transition). The latter need explanation which follows.

The language of predicate logic has been designed to express natural language sentences formally and unambiguously. This was done in a biased way since many of those involved in the design (such as Frege [Frege, 1879]) had mainly sentences of a mathematical nature in mind. Sentences involving actions were not taken into serious consideration until the publication of the situation calculus in 1969 [McCarthy and Hayes, 1969] in which any n-ary relation *P* is extended to an (n + 1)-ary one by an argument for determining the situation in which the relation is meant to hold (see Section 5.4).

Natural language apparently does not need such an extra vehicle. A (static) mathematical sentence (such as "if a number is greater than zero then it is positive") looks exactly like a (dynamic) one about actions (such as "if I take the book then it is mine"). In [Bibel, 1986a] the main idea for a logic has been outlined which resembles natural language more closely in this aspect of treating actions than does the situation calculus. The approach then was called *linear connection method* or shortly LCM; for the logic we introduce here on the basis of LCM we propose the name *transition logic* or TL. The idea underlying LCM spawned a great number of studies based on this idea such as [Fronhofer, 1987; Bibel *et* a/., 1989; Holldobler and Schneeberger, 1990; Grofie *et* a/., 1992; Bruning *et* a/., 1992; 1993; Holldobler and Thielscher, 1993; 1995; Grofie *et* a/., 1996; Fronhofer, 1996; Herrmann and Thielscher, 1996; Thielscher, 1996; Eder *et* a/., 1996; Thielscher, 1997b; Bornscheuer and Thielscher, 1997; Thielscher, 1997a] to mention several of them. Here facts may be treated as resources which may be consumed by actions. Two different formalisms are used to achieve this. One, TL, employs the additional set of resource-sensitive operators &, |, $\Rightarrow$ just introduced (the other achieves their effects on the term level of classical logic as we will see in Section 5.3).

A rule $K \Rightarrow L$, called an *action* (or *transition)* rule (or *effect axiom)*, models an action which consumes *K* and produces *L.* For instance,

$$on\_table(book) \Rightarrow in\_hand(book)$$

can be seen as the equivalent in TL of the situation calculus rule

$$on\_table(book, s) \rightarrow in\_hand(book, result(take, s)).$$

In classical logic $L \wedge L$ is equivalent with *L* according to the rule of idempotence. In real-world scenarios it does matter, however, whether you have the same thing (say a dollar) once or twice. Similarly, it does matter whether you take your dollar or mine. That is why we need the two extra operators &, | which behave just like their classical counterparts $\wedge, \vee$ except for the rule of idempotence, which does *not* hold for them, and for | modelling an exclusive (rather than an inclusive) alternative. In consequence, we will not have the law of distributivity which allows | to be distributed over &.

Formulas built from literals by means of the quantifiers and the resource-sensitive operators only are called *r-formulas,* r-formulas without | are also called *conjunctive* r-formulas. General *formulas* of TL are r-formulas, and any expression built from those by means of the classical operators. For instance, $P \,\&\, (P \Rightarrow Q) \Rightarrow Q$ is an r-formula, hence a formula, $(P' \rightarrow P) \rightarrow [P' \,\&\, (P \Rightarrow Q) \Rightarrow Q]$ is a formula but not an r-formula, and $P \,\&\, (P \rightarrow Q) \Rightarrow Q$ is not a formula (nor an r-formula) since the definition does not allow classical operators (other than quantifiers) below a resource-sensitive one in the formula tree. An r-subformula which is not a proper

subformula of an r-subformula is also called an *r-part* in the given formula.

S Basic deductive machinery

We will deal in this paper with a restricted class of r-formulas only which have the form $G_0 \& (F_1 \Rightarrow G_1) \& \ldots \& (F_n \Rightarrow G_n) \Rightarrow H$ whereby $\Rightarrow$ does not occur in F_i, G_j, H. Semantic entailment $\models$ for the resulting class of formulas will be introduced only informally. $T \rightarrow [K \& (K \Rightarrow L)] \models F$ holds if *F* is classically entailed by $T \rightarrow K$ or by $T \rightarrow L$, depending of the state reached by not performing or performing the transition $K \Rightarrow L$ in the r-part which, if executed, consumes *K* and produces *L*.

As we see two different states, say σ_0, σ_1, are to be distinguished in this example, the one before and the other after the transition. Semantic entailment is dependent on these states. For instance, we might write $T \rightarrow [K \& (K \Rightarrow L)]_{\sigma_0} \models T \rightarrow K$. As more transitions get involved we obtain more such states to be distinguished.[1] Validity, $\models F$, is then defined as usual. Section 5 will resume the discussion of the semantics of TL while in the present section we focus on its deductive aspects.

As the original name of our approach, *linear connection method* (LCM), suggests, the basic deductive machinery is based on the connection method [Bibel, 1993; 1987]. This deductive method is characterized by the fundamental theorem which in turn characterizes validity of a formula by the so-called spanning property explained shortly. Many different logical calculi can be based on this method.

In order to explain the spanning property we need the concepts of a path through a formula and of a connection. A *path* through a formula *F* is the set of literals of any conjunct of the conjunctive normal form of *F*. Paths can best be illustrated if formulas are displayed as matrices. Matrices (positively) represent disjunctions of clauses which in turn represent conjunctions of literals (or, in general, matrices). Consider the formula $P \wedge (P \rightarrow Q) \rightarrow Q$ (expressing the well-known logical rule of modus ponens). In negation normal form the same formula reads $\neg P \vee (P \wedge \neg Q) \vee Q$, which is a disjunction of three clauses. Hence as a matrix it looks as follows.

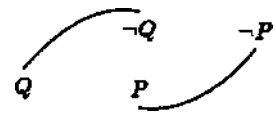
$$\begin{bmatrix} & \neg Q & \neg P \\ Q & P & \end{bmatrix}$$

As an aside we mention that a rotation of this matrix by 90° (counterclockwise) basically yields the corresponding PROLOG program except for the differences due to the negative representation used in PROLOG.

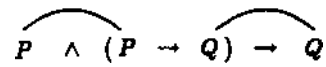
```
P.
Q   :-  P.
    ?-  Q
```

A path through such a matrix (or the formula it represents, or the corresponding PROLOG program) is now the set of literals obtained by selecting exactly one literal from each clause (or, in other words, traversing the matrix say from left to right). In the present example there are exactly two such paths, namely $\{Q, \neg Q, \neg P\}$ and $\{Q, P, \neg P\}$. The disjunction of the literals of these two paths are obviously the disjuncts of the conjunctive normal form of $\neg P \vee (P \wedge \neg Q) \vee Q$.

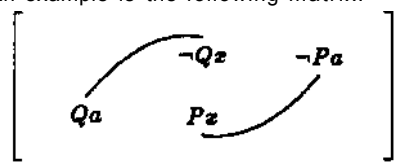
A *connection* is a subset of a path of the form {R, -R}. There are two connections in our present example illustrated as arcs in the following display.



A set of connections (or *mating)* is called *spanning* if each path through the matrix contains at least one connection. This is the case for the two connections of our example, hence the formula is (of course) valid according to the fundamental theorem mentioned at the outset of the section. Recall that the matrix form is used just for illustration and is thus not essential for the connection method. The connections (and the spanning property) could as well have been identified in the original formula as follows.



A chain of two (or more) connections like the two displayed in the matrix may thus be regarded as an encoding of one (or more) applications of modus ponens. This illustration also demonstrates that it is connections which lie at the heart of deductive reasoning (more so than rules like the $P \rightarrow Q$ in the example). In some sense a connection may also be seen as an encoding of an application of the well-known resolution rule. So far connections have been illustrated for propositional examples. They apply to first-order formulas in the obvious way, connecting literals with opposite signs and unifiable terms. An example is the following matrix.

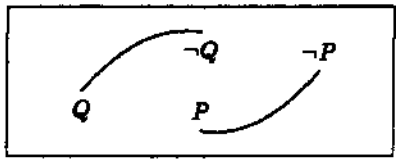


Here validity is established by the two spanning connections along with the substitution *x\a,* which makes the connected hterals complementary.

Up to this point we have restricted our discussion to deduction for purely classical formulas. The characterising spanning property carries over to the case of general

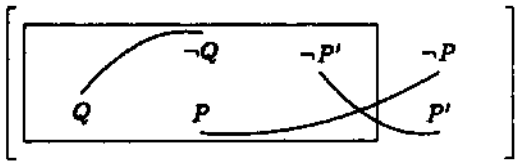
[1] [Thielscher, 1997a] gives a precise semantics which, however, needs adaption to TL and the view just outlined.

formulas in our logical language with one minor modification to be explained shortly. In fact, if we take the r-formula $P \,\&\, (P \Rightarrow Q) \Rightarrow Q$ as our first example then we may use exactly the same matrix as the one before to represent the formula in a two-dimensional way. In fact, in spite of the modification in the formula exactly the same proof for validity is obtained.



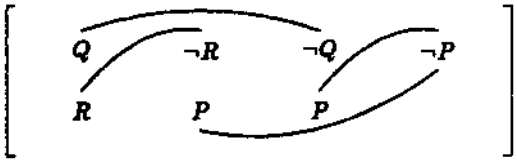
In order to distinguish it from the classical matrix we use a box rather than brackets. One should note, however, that the semantics of the operations represented by the structural arrangement this time is quite a different one. As in the classical case the matrix representation is more of an illustrative relevance, since the connections also here could as well have been placed in the original formula.

Let us now consider a general formula like $(P' \rightarrow P) \rightarrow [P' \,\&\, (P \Rightarrow Q) \Rightarrow Q]$ which in its classical part expresses that P' is just another name for P and which is represented by the following matrix.

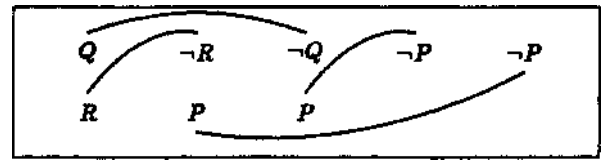


The r-submatrix is boxed.[2] The three connections are obviously spanning. Hence the formula is again valid.

The need for a modification becomes clear if we compare the classically valid formula $P \wedge (P \rightarrow Q) \wedge (P \rightarrow R) \rightarrow Q \wedge R$ whose proof is



with the analogical r-formula $P \,\&\, (P \Rightarrow Q) \,\&\, (P \Rightarrow R) \Rightarrow Q \,\&\, R$. While the validity of the first formula is clear from the spanning property obviously satisfied for the matrix, the second formula should intuitively *not* be valid. Namely, if a dollar *(P)* buys a coffee (Q), and if a dollar buys a tea *(R),* and if I have just one dollar (as the formula suggests) then clearly I cannot buy both coffee and tea since I would rather need *two* dollars for that purpose. Since the matrix and its connections would be exactly the same for the second formula as for the first one, we are lead to conclude that the spanning property (characterising validity in the classical case) needs some modification. The kind of modification becomes clear if we add another P to the present r-formula, ie. $P \,\&\, P \,\&\, (P \Rightarrow Q) \,\&\, (P \Rightarrow R) \Rightarrow Q \,\&\, R$, which intuitively is valid as just illustrated and compare its proof



with the previous one. In the latter matrix *each literal is contained in at most one connection* while in the former this *linearity restriction in its original form* [Bibel, 1986a]) is not satisfied because the literal $\neg P$ is contained in more than one, namely in two connections. To cover the general case considered in the present paper this linearity restriction needs a more general definition.

For that purpose we inductively introduce the concept of the *directionality*[3] 0 (for consumption) and 1 (for resource) of the nodes in the formula tree of an r-formula. The root has directionality 0. If a node with directionality d is labelled by & or by | then its successor nodes have the same directionality d; if it is labelled $\Rightarrow$ then the directionality of the left successor node is (d + 1) mod 2 while that of the right successor node is d. The directionality partitions the occurrences of literals in an r-formula (or r-matrix) into *resource literals* if their directionality in the formula is 1, and *consumption literals* if it is 0. We attach this directionality to a literal if needed as an upper index. In all our matrix examples the directionality is 1 for a negated literal and 0 for an unnegated one.

A *chain of connections* in a matrix M is a sequence $(c_1, \ldots, c_n)$, $n \geq 1$, of directed connections $(\overline{L}_i, L_i)$, such that for any i, $1 \leq i \leq n-1$, the *end literal* of c_i, ie. L_i, and the *start literal* of c_{i+1}, ie. $\overline{L}_{i+1}$, are literals in the same (top level) clause of M but do not occur in one and the same path through M, ie. they are *vertically related* [Fronhöfer, 1996]. A *cycle* is a chain such that also its *start literal* $\overline{L}_1$, ie. the start literal of c_1, and its *end literal* L_n are vertically related in this sense. A chain is called an *r-chain* if, viewed as a list $(K_1, \ldots, K_{2n})$ of literals K_j, $1 \leq j \leq 2n$, $n \geq 1$, K_1 is of directionality 1, K_1 and K_{2n} are in the same r-part M_0 of M while no other K_k, $k \neq 1, 2n$ is in M_0, and K_{2i} and K_{2i+1} are in different r-parts. Two r-chains $(K_1, \ldots, K_{k-1}, K_k, \ldots)$ and $(K_1, \ldots, K_{k-1}, K'_k, \ldots)$ are called *c-distinct* if K_k and K'_k occur in different r-parts of M, or $k-1$ is odd and K_k and K'_k are different literal occurrences. A set of connections in a matrix M *satisfies the linearity restriction* if each directionality-1 literal in an r-part of M is start literal of no more than one c-distinct r-chains.

These definitions go a bit beyond what is intended with this paper simply because they are novel and have not been published before. In fact it could well be that they need further adjustment once the structure of formulas and their semantics are finally settled. We just mention here that the definitions aim at yielding the

[2]For simplicity we do not box the literals, which formally are r-parts, in the classical part.

3Note that directionality is not the same as polarity.