2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Using Constrained Intuitionistic Linear Logic for Hybrid Robotic Planning Problems

Uluç Saranlı and Frank Pfenning

***Abstract*— Synthesis of robot behaviors towards nontrivial goals often requires reasoning about both discrete and continuous aspects of the underlying domain. Existing approaches in building automated tools for such synthesis problems attempt to augment methods from either discrete planning or continuous control with hybrid elements, but largely fail to ensure a uniform treatment of both aspects of the domain. In this paper, we present a new formalism, Constrained Intuitionistic Linear Logic (CILL), merging continuous constraint solvers with linear logic to yield a single language in which hybrid properties of robotic behaviors can be expressed and reasoned with. Following a gentle introduction to linear logic, we describe the two new connectives of CILL, introduced to interface the constraint domain with the logical fragment of the language. We then illustrate the application of CILL for robotic planning problems within the Balanced Blocks World, a "physically realistic" extension of the Blocks World domain. Even though some of the formal proofs for the semantic foundations of the language as well as an efficient implementation of a theorem prover are yet to be completed, CILL promises to be a powerful formalism in reasoning within hybrid domains.**

I. INTRODUCTION

One of the central goals of research in robotics and artificial intelligence has been to build systems that are capable of autonomous operation towards higher level goal specifications. In this context, a number of different approaches have been considered, adopting methods from either a discrete planning perspective based on action sequences [9, 20, 22], or from control theory and continuous mathematics [19]. Beyond these two extremes, hybrid approaches have also been tried to reach the scalability and generality required for successful robot operation in unstructured environments [3, 8, 13], but a uniform framework for the representation and solution of such problems still remains elusive. Most often, attempts to combine discrete reasoning with continuous control have been through the discretization of underlying continuous spaces and the application of purely discrete methods such as combinatorial search, model checking [6] and logical reasoning [7] to the resulting approximation [1]. These approaches often run into either computational complexity problems due to the size of the resulting discrete state space, or performance problems due to approximations introduced during the discretization of continuous properties.

In this paper, we describe a new representational framework based on intuitionistic linear logic [4, 11, 18], extended with two new logical connectives to incorporate continuous constraints directly into the language. Traditional linear logic provides a tractable way of representing and reasoning with properties of changing state, effectively overcoming the frame problem [15], which has been one of the central issues in the use of logical formalisms for robotic planning problems. Linear logic treats hypotheses as *consumable resources*, providing a natural way of representing the effects of actions on the state of a robot and its environment. Classical features of the environment are then recovered by the use of so-called "unrestricted" hypotheses, encoding invariant facts about the environment and the robot's behaviors.

Unfortunately, in its original form, linear logic cannot reason effectively about continuous aspects of a robot's behavior. Even though certain domains such as real inequalities can be encoded within purely logical theories, the resulting complexity is large enough to significantly limit their practical applicability. This is the primary motivation behind our extensions, combining the efficiency of domain specific constraint solvers with the expressive power of linear logic within a computationally tractable proof theory.

Existing work on combining intuitionistic linear logic with a classical constraint domain includes ILC [12], which is closely related to our system. In ILC, classical formulas are isolated by a modal operator rather than binary connectives as we do here. Furthermore, Jia and Walker provide a specific semantics of their logic to describe the structure of heaps in an imperative programming language, which is quite different from the mix of discrete and continuous reasoning we aim at here. Going back further, the first proposal to add constraint domains to linear logic that we are aware of is by Bozzano [2], who uses them to describe and reason about distributed computing systems and protocols. His setting is significantly more restrictive than ours.

In the rest of the paper, we first briefly present the theoretical foundations of Constrained Intuitionistic Linear Logic (CILL) and the associated proof theory, only covering details necessary to understand the underlying ideas and their applications to robotic planning problems. We then introduce the Balanced Blocks World (BBW), a physically realistic extension of the popular blocks world domain. Finally, we present how problems in BBW can be represented within CILL and how logical proofs can be transformed into plans with formal physical and logical validity properties.

II. CONSTRAINED INTUITIONISTIC LINEAR LOGIC

Formal specification of deductive systems can be accomplished in a number of different ways. In formalizing CILL, we will find it useful to adopt a Gentzen style formulation,

U. Saranlı is with the Dept. of Computer Engineering, Bilkent University, Bilkent, 06800 Ankara, Turkey saranli@cs.bilkent.edu.tr
F. Pfenning is with the Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA fp@cs.cmu.edu

based exclusively on inference rules. This approach yields a formalism which not only highlights the very useful isomorphism between proofs and programs for intuitionistic settings but is also very suitable for efficient proof search.

In order to formulate CILL using *sequent calculus* [10], we first introduce a *CILL sequent* as $\Psi \mid \Gamma; \Delta \implies A$. The intended meaning of this judgement [14] is that if the constraints in Ψ are satisfiable, then we can achieve the goal A using the unrestricted hypotheses Γ and the linear resources in Δ. Even though this kind of definition is fairly standard in theoretical computer science, we will find it useful to explain some of the key concepts in this definition for the benefit of the robotics community.

First and foremost, we assume that a first order, possibly classical constraint domain from which the constraint expressions in Ψ are drawn is specified. This domain is expected to possess both formal properties such as a consistent and complete notion of *entailment* as well as a decision procedure capable of solving sets of constraint expressions in an efficient manner.

A very important detail in the definition of this sequent is the distinction between unrestricted hypotheses and linear resources, which is particular to and characteristic of linear logic. The former includes hypotheses which can be used as many times as necessary (or not at all), while the latter includes resources that must be used *exactly once*. Even though these two sets of formulae have the same syntax, they are distinguished in the sequent so that the formulation of the proof theory can give them the desired meaning through an appropriate choice of inference rules.

The last key property of our formulation is that it is *intuitionistic* (or *constructive*) in nature. As such, the meaning of a judgement is defined by the evidence given in its support, i.e. its proof. In contrast to *classical* logic which relies on an external notion of truth to define the meaning of judgements, intuitionistic formulations admit much more immediate interpretations of proofs as plans, nicely serving our ultimate goal of constructing executable plans for robot systems.

This choice, however, also introduces important issues. Most importantly, since our primary goal is to ensure that certain physical properties hold for constructed plans, there is inherent, inevitable external semantics that must be admitted by the logic. However, it is often quite difficult to establish a formal correspondence between the so-called *proof-theoretic* semantics and externally imposed *denotational* semantics for intuitionistic constructions, especially in the case of linear logic. Even though we do not explicitly address this issue in the present paper, this connection is one of the important elements in ensuring the correctness of CILL based plans with respect to the physical meaning of the stated goals.

A. The Unconstrained Fragment

CILL inherits all the connectives from intuitionistic linear logic, summarized in the first part of Table I. In this section, we summarize the meanings of these connectives by giving the associated sequent calculus rules of inference.

existing connectives	
$\otimes$	simultaneous conjunction
$\multimap$	linear implication
$\mathbf{1}$	unit
$\&$	alternative conjunction (internal choice)
$\top$	top
$\oplus$	disjunction (external choice)
$\mathbf{0}$	impossibility
$\supset$	unrestricted implication
$!$	of course
new connectives	
$\supset_c$	constrained implication
$\wedge_c$	constrained conjunction

TABLE I
SUMMARY OF CILL CONNECTIVES

In contrast to the single conjunctive connective $\wedge$ of classical logic, linear logic features two different connectives: The simultaneous conjunction $\otimes$ and the alternative conjunction &. The former represents cases where two goals are simultaneously realizable in the same state, requiring available resources to be properly split into two. In contrast, the latter captures the idea that currently available resources are sufficient to individually achieve either goal, but not both of them at the same time. The following inference rules, (intended to be read from the bottom to the top), illustrate the distinction between these two types of conjunction.

$$\frac{\Psi \mid \Gamma; \Delta_1 \implies A \qquad \Psi \mid \Gamma; \Delta_2 \implies B}{\Psi \mid \Gamma; \Delta_1, \Delta_2 \implies A \otimes B} \otimes R$$

$$\frac{\Psi \mid \Gamma; \Delta \implies A \qquad \Psi \mid \Gamma; \Delta \implies B}{\Psi \mid \Gamma; \Delta \implies A \,\&\, B} \&R$$

The key distinction is that in the $\otimes R$ rule, the resources in the goal sequent are divided among the two subgoals (as Δ_1 and Δ_2), whereas in the $\&R$ rule, the goal resources are duplicated into both subgoals.

In logical systems, causality is usually captured through *implication*. However, since classical implication allows unlimited use of the premise[1], linear logic needs a revised connective. *Linear implication*, $\multimap$ ensures that available resources are used exactly once, both for the premise and for the conclusion. Associated sequent calculus rules are given as

$$\frac{\Psi \mid \Gamma; \Delta, A \implies B}{\Psi \mid \Gamma; \Delta \implies A \multimap B} \multimap R$$

$$\frac{\Psi \mid \Gamma; \Delta_1 \implies A \qquad \Psi \mid \Gamma; \Delta_2, B \implies C}{\Psi \mid \Gamma; \Delta_1, \Delta_2, A \multimap B \implies C} \multimap L \quad .$$

Consider the rule $\multimap R$. In order to prove that an implication holds, it is necessary and sufficient to prove the conclusion B by adding the premise A to the list of available resources. The left rule, on the other hand, captures how one would go about using an implication as one of the available resources. This rule is particularly important for our domain since we use linear implication to capture *actions*, with preconditions in the premise and postconditions in the conclusion. In using $\multimap L$, we have to first prove the subgoal

[1]For example, the formula $A \rightarrow A \wedge A$ is valid in classical logic

that we can achieve A using part of the resources in the overall goal, and then we will be allowed to use B as one of our resources in proving C in the second subgoal. This ensures that the preconditions are met before we can proceed by assuming the postconditions of an action.

Another important component is Top, $\top$, which is analogous to "truth" in intuitionistic logic. As a goal, it consumes all available resources, illustrated by the rule

$$\frac{}{\Psi \mid \Gamma;\, \Delta \implies \top}\ \top R\ .$$

The presence of $\top$ on the left side gives no additional information. Consequently, having $\top$ as a hypothesis has no utility and hence it has no left rule.

Finally, in order to obtain linearity and ensure that unrestricted hypotheses are properly incorporated into the system, two "hypothesis" rules are needed:

$$\frac{}{\Psi \mid \Gamma;\, A \implies A}\ init \qquad \frac{\Psi \mid (\Gamma, A);\, (\Delta, A) \implies A}{\Psi \mid (\Gamma, A);\, \Delta \implies A}\ copy$$

The first rule guarantees that all linear resources are used, while allowing leftovers in the set of unrestricted hypotheses. In conjunction with the rules for individual connectives, this properly implements linearity as defined before. The *copy* rule, on the other hand, allows shifting formulae from the set of unrestricted hypotheses to the set of linear resources, establishing a connection between the two while allowing multiple uses of the same unrestricted hypothesis.

For the sake of brevity, we will omit the rules for the remaining connectives, but the interested reader can refer to various resources in the literature for further details. Our formulation is closest in spirit to the approach taken in [4].

B. New CILL Connectives

All inference rules associated with traditional connectives directly copy the constraint context Ψ from the goal to the subgoals. Consequently, they do not offer any way to couple constraint expressions with specific parts of the reasoning process. Constraints on the final goal are the same as the constraints for all subgoals, eliminating any possibility of capturing different physical constraints on individual actions (i.e. hypotheses).

The primary contribution of our work is a general framework in which such constraint expressions can be injected into linear logical formulae by the use of two new connectives in addition to those described in Section II-A. Syntactically, both of these newly introduced binary connectives merge a constraint expression with a linear formula. The first one, the constraint implication $\supset_c$, introduces a constraint precondition to a linear expression. In order to achieve $D \supset_c A$, we need to show that the goal A can be achieved under the conjunction of existing constraints Ψ and the guard D.

$$\frac{(\Psi, D) \mid \Gamma;\, \Delta \implies A}{\Psi \mid \Gamma;\, \Delta \implies D \supset_c A}\ \supset_c R$$

The left rule for the guard connective is very similar to typical left rules for implication, except that the proof for the guard constraint D is now handled by a separate proof procedure that is specific to the constraint domain.

$$\frac{\Psi \models D \qquad \Psi \mid \Gamma;\, \Delta, A \implies C}{\Psi \mid \Gamma;\, \Delta, D \supset_c A \implies C}\ \supset_c L$$

The second operator we introduce is the constraint conjunction $\wedge_c$, which asserts the validity of a constraint in conjunction with a linear logic expression. The right rule in this case is very similar to a typical intuitionistic conjunction except that the validity of the constraint D is pushed to a branch that is handled by a domain specific constraint solver.

$$\frac{\Psi \models D \quad \Psi \mid \Gamma;\, \Delta \implies A}{\Psi \mid \Gamma;\, \Delta \implies D \wedge_c A}\ \wedge_c R$$

For the constraint conjunction left rule, we use an approach reminiscent of $\otimes L$, wherein the goal C needs to be achieved with augmented constraints and linear hypotheses.

$$\frac{(\Psi, D) \mid \Gamma;\, (\Delta, A) \implies C}{\Psi \mid \Gamma;\, \Delta, D \wedge_c A \implies C}\ \wedge_c L$$

As it can easily be seen from these inference rules, we now have a way in which we can interleave constraint expressions with linear formulae, together with a proof system that is capable of separating reasoning in the constraint domain from reasoning within the linear fragment of the language. By using quantifiers whose elements are drawn from the constraint domain, communication between the two fragments becomes possible. The logical behavior of such quantifiers and variables are in concordance with their usual rules, although their presence (and, in particular, sequences of alternating quantifiers) can have a dramatic impact on the tractability of the constraint solving problem associated with proof search.

In our examples, variables shared between the constraint and logical fragments allow using constraint implication as a "guard" for action preconditions and the constraint conjunction as an assertion of continuous properties of action postconditions.

C. Formal Properties of CILL

An important property to be established for deductive systems formalized through sequent calculus is the ability to eliminate explicit uses of the *cut rule*, which make it very difficult to construct efficient proof search mechanisms. For CILL, there are two such rules,

1) If $\Psi \mid \Gamma;\, \Delta \implies A$ and $\Psi \mid \Gamma;\, (\Delta', A) \implies C$, then $\Psi \mid \Gamma;\, (\Delta, \Delta') \implies C$,
2) If $\Psi \mid \Gamma;\, \cdot \implies A$ and $\Psi \mid (\Gamma, A);\, (\Delta) \implies C$, then $\Psi \mid \Gamma;\, (\Delta) \implies C$,

both of which turn out to be admissible in the presence of the newly added connectives and their associated rules of inference. Under suitable assumptions about logical entailment for the constraint domain, the admissibility proof is fairly straightforward and follows the methods detailed in [17].

The second, perhaps more fundamental issue is the connection of the proof theory outlined in preceding sections to the desired external semantics for CILL formulae. Even though we have not yet proven that the system is complete with respect to physically meaningful, externally imposed semantics, it is clear that most of the problems in establishing such a property arise within constraint domains that are not convex[2]. In those cases, we are faced with the problem of *case splitting*, which requires details of the constraint domain to affect proof search [16]. We believe that the inclusion of an additional inference rule in the form

$$\frac{\Psi \models A \vee B \qquad \Psi, A \,|\, \Gamma;\, \Delta \implies C \qquad \Psi, B \,|\, \Gamma;\, \Delta \implies C}{\Psi \,|\, \Gamma;\, \Delta \implies C}$$

will be necessary to ensure completeness of the overall system. However, we leave a more detailed treatment of this and other associated issues outside the scope of this paper.

III. The Balanced Blocks World

A. Motivation and the Domain

Since the very early days of AI, the Blocks World domain served as a simple but reasonably rich testbed for planning algorithms and methods [21]. Unfortunately, as it has often been the case within the discrete planning community, the physical relevance of the abstractions on which various planning formalisms are built received little attention. Consequently, even though significant progress has been made in advancing the state of the art with respect to the efficiency and optimality of constructed plans, the wide gap between purely discrete toy problems and complex hybrid systems is yet to be tackled in a uniform and principled way.

In order to illustrate the application of CILL to robotic planning problems, we introduce the Balanced Blocks World (BBW), in which dynamic balance and physical alignment properties of planar blocks are also considered in conjunction with logical properties associated with different stackings.

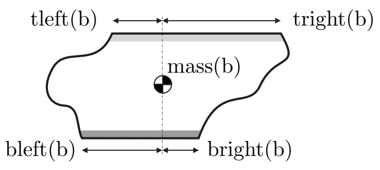

Fig. 1. Static parameters of a block in the *Balanced Blocks World*

Each block in BBW is named and their geometry is specified through five functions from the set of block names to real numbers, illustrated in Figure 1. Furthermore, in order to enrich the logical fragment of the planning problem, the top and bottom surfaces of each block are assigned specific colors, intended to capture "compatibility" between blocks.

The current state of block placements is captured through a number of predicates, summarized in the top half of Table II. These predicates are intended to be used as linear resources to capture the dynamic nature of the system state. In contrast, the predicates defined in the bottom half of Table II will be used to capture invariant facts about the world such as the colorings of individual blocks and slot positions.

[2]As in [16], we call a constraint formula non-convex if it entails a disjunction of constraints without entailing any of the constraints alone.

dynamic state of the system	
tableempty(i)	There are no blocks on slot i of the table
ontable(b, i)	Block b is directly on slot i of the table
available(b)	Block b is available for placement
on(a, b, x)	Block a is on top of block b at an absolute position x
clear(b, x)	Block b is at absolute position x and its top is clear
invariant facts about the world	
tcol(b, c)	The top of block b has color c
bcol(b, c)	The bottom of block b has color c
slotcol(i, c)	Slot i on the table has color c
slotisat(i, x)	Slot i is located at distance x from table origin

TABLE II

Resource predicates for the Balanced Blocks World

B. Basics: Using CILL for Planning in the BBW Domain

As in [18], we model the starting state for the planning problem as a set of linear propositions. For instance, a very simple state where there is a single empty slot on the table and two available blocks yields the linear context

$$\Delta_o = (\text{tableempty}(1), \text{available}(\mathbf{a}), \text{available}(\mathbf{b}))\ .$$

In contrast, the unrestricted context Γ serves two purposes. First, it includes logical formulae to capture invariant facts about the environment. For the example above, we have

$$\begin{aligned}\Gamma_f \;\;=\;\; & (\text{tcol}(\mathbf{a}, red), \text{bcol}(\mathbf{a}, blue), \text{tcol}(\mathbf{b}, blue), \\ & \text{bcol}(\mathbf{a}, grn), \text{slotcol}(1, grn), \text{slotisat}(1, 0)\ .\end{aligned}$$

Second, the unrestricted hypotheses include models of actions that are available in the domain in the form of linear implications, summarized in Table III. For example, the hypothesis labeled **putontable** models the act of placing an available block onto an empty slot on the table. Note that the linear resources associated with the current state of the system are "consumed" and new resources are introduced to indicate the updated state of the system. This action does not involve any constraint connectives since slots on the table are expected to support blocks of any size and shape. In contrast, the stacking of blocks on top of each other requires that balance constraints are satisfied (see Section III-C).

The final necessary component in specifying the planning problem is a goal. Since we adopt an intuitionistic setting, this is required to be a single linear formula, appearing on the right side of the sequent. If, for instance, our goal is to reach a state where block $\mathbf{b}$ is placed either on the table or on another block, this would be expressed as

$$G = (\exists i.\ \text{ontable}(\mathbf{b}, i) \oplus \exists a.\ \exists x.\ \text{on}(\mathbf{b}, a, x)) \otimes \top$$

in which the disjunction connective is used to indicate that the plan construction has to pick which one of the two alternatives will be satisfied. $\top$ is used to specify incomplete goals since it consumes all resources left unused by the rest of the proof. At this point, the planning problem reduces to finding a proof for the sequent

$$\Psi_e \,|\, (\Gamma_f, \Gamma_a);\, \Delta_0 \implies G$$

where additional environmental constraints can be specified in Ψ_e. Such a proof will include a sequence of applications of the $\multimap L$ rule on hypotheses corresponding to actions, which can be easily converted to an executable plan.

putontable	$: \forall a.\ \forall i.\ \forall c.\ \forall x_i.\ \text{available}(a) \otimes \text{tableempty}(i) \otimes \text{slotcol}(i,c) \otimes \text{bcol}(a,c) \multimap \text{ontable}(a,i) \otimes \text{clear}(a)$
getofftable	$: \forall a.\ \forall i.\ \text{ontable}(a,i) \otimes \text{clear}(a) \multimap \text{available}(a) \otimes \text{tableempty}(i)$
putonblock	$: \forall a.\ \forall b.\ \forall c.\ \exists x_a.\ \text{available}(a) \otimes \text{clear}(b) \otimes \text{bcol}(a,c) \otimes \text{tcol}(b,c) \multimap \text{on}(a,b,x_a) \otimes \text{testing}(a) \otimes \text{check}(b, mass(a), x_a)$
getoffblock	$: \forall a.\ \forall b.\ \exists x_a.\ \text{on}(a,b,x_a) \otimes \text{clear}(a) \multimap \text{available}(a) \otimes \text{clear}(b)$
checkiter	$: \forall a.\ \forall b.\ \forall m.\ \forall x_m.\ \forall x_a.\ \text{check}(a,m,x_m) \otimes \text{on}(a,b,x_a) \multimap$ $isin(x_m - x_a, \text{tleft}(a), \text{tright}(a)) \supset_c \left(\text{check}(b, m + mass(a), \frac{m x_m + mass(a) x_a}{m + mass(a)}) \otimes \text{on}(a,b,x_a)\right)$
checkend	$: \forall a.\ \forall b.\ \forall m.\ \forall i.\ \forall x_a.\ \forall x_m.\ \text{check}(a,m,x_m) \otimes \text{ontable}(a,i) \otimes \text{slotisat}(i,x_a) \otimes \text{testing}(b) \multimap$ $isin(x_m - x_a, \text{tleft}(a), \text{tright}(a)) \supset_c (\text{ontable}(a,i) \otimes \text{clear}(b))$.

TABLE III

CILL REPRESENTATIONS OF BBW ACTIONS AND SUPPORTING RULES FOR CHECKING BALANCE OF NEWLY PLACED BLOCKS.

C. A Detailed Example: Stacking A Single Tower

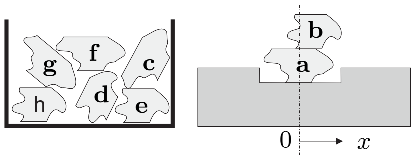

Fig. 2. Stacking a single tower from a set of available blocks

One of the simplest BBW goals we can represent in CILL is that of stacking a single tower of blocks to reach a certain minimum distance away from the table origin using available blocks (see Figure 2). Together with coloring constraints, this represents a hybrid goal, difficult to capture within traditional formalisms. More formally, given a constant x_g, we would like to achieve the goal

$$G := \exists a.\ \exists b.\ \exists x.\ (x > x_g) \wedge_c \text{on}(b,a,x)$$

from an arbitrary initial condition.

One of the interesting features of the BBW domain is exposed by the primitive action **putonblock** to place a block on top of an existing one on the tower. To keep the tower balanced, it is not sufficient to check that the center of mass of the new block rests on the top supporting surface of the tower. We also need to check that the newly placed block does not cause any part of the existing structure to collapse. Even though this problem is easier to solve if we were to start from the top of the stack and plan backwards, we would like to constrain our action representations to move the system forward in "time" [3] in order to admit more complex goals such as bridges between two adjacent towers or additional concepts such as stability margins of constructions. Consequently, the primary action in our encoding is that of placing a block on top of an existing block, encoded with the hypothesis labeled **putonblock** in Table III.

To perform this balance check, we recursively compute the center of mass of connected groups of blocks from top to bottom and ensure that all combined centers of mass lie on support surfaces. This is accomplished by using the resource check(a, m, x_m), intended to mean that a group of blocks with combined mass m and center of mass x_m will stay balanced on block $\mathbf{a}$. This predicate is introduced for the topmost block with the application of the **putonblock**, iterated through the hypothesis **checkiter** until the query reaches the bottom of the block with the hypothesis **checkend**.

[3]For the time being, CILL's concept of time is limited to forward progress and does not capture quantitative aspects of temporal relations.

Note that by delaying the generation of clear(c) while the balance check is being performed, we effectively halt the execution of any more block placements on the same tower until the end of the recursive balance check marked by the termination rule above. The predicate $isin()$ is simply used as a shorthand for a conjunction of linear inequalities.

Even this relatively simple goal illustrates the expressive power of CILL in incorporating logical reasoning (i.e. different orderings of blocks) and continuous constraints (i.e. balance requirements) towards a goal specified in terms of satisfying a specific continuous constraint.

D. Examples on More Complex Planning Problems

The example presented in Section III-C does not significantly exercise logical reasoning, but relies more heavily on the constraint domain, often reducing to a constrained combinatorial search. In this section, we briefly describe more complex scenarios in which the logical fragment of the formalism would be expected to play a more significant role in pruning the tree of possible action sequences.

1) Multiple towers and no free use of a "bucket": The typical blocks world planning problem includes an initial state with multiple existing stacks of blocks and a desired final state consisting of a particular ordering. This more constrained domain usually adopts a single "buffer" in the form of a robot which picks up a block and places it either on the table or on top of an existing tower. Consequently, plans can no longer make free use of an unlimited buffer (the bucket), imposing stricter constraints on the set of possible plans.

A similar setting in BBW would place more burden on the logical fragment of the reasoning, enabling more goal-directed pruning of the search over possible actions. Along similar lines, one can also explore a range of possibilities between underspecified goals such as the one in the previous section and a fully specified goal states through the use of existential quantifiers. The more aspects of the goal are specified, the better chance the proof engine has to proceed in a goal directed manner, emphasizing the advantages of representing the problem within a logical formalism.

2) Bridges across towers: Suppose, for example, that we were working with two slots on opposite ends of a table, located at $x = -1$ and $x = 1$. A possible goal would be to place a block at $x = 0$, but the available blocks are not sufficient to build a single tower capable of reaching this goal. However, if we introduce a very wide block, together with a new action to place long blocks as a bridge between two towers, this would admit a richer set of constructions, admitting a feasible solution in this case.

There are, of course, further extensions and more complex examples that are possible within the BBW setting. However, we hope that the examples given above concisely illustrate the expressive power of our formalism and its application for planning problems in hybrid domains.

IV. CONCLUSION

In this paper, we introduced a new logical formalism, Constrained Intuitionistic Linear Logic, in which it is possible to express and reason with hybrid properties of systems which incorporate both discrete and continuous elements. The distinguishing feature of our approach is that instead of artificially mapping one of these elements into an unnatural representation within the other, we merge available reasoning tools for both within a single, uniform formalism. Doing so allows the use of efficient decision procedures for constraint domains in conjunction with theorem proving in linear logic to yield an effective method for constructing behavioral plans for physically realistic domains. The expressive power of the formalism and its application for physically realistic robotic planning problems were illustrated in the Balanced Blocks World, an example domain incorporating physical models of balance into the familiar toy domain of Blocks World.

Some of the theoretical foundations of CILL still need to be finalized. In particular, a well defined semantics and the completeness of the proof theory must be established. Especially in the presence of non-convex constraint domains, there are significant challenges in doing this while preserving the ability to perform efficient proof search and maintaining the elegant separation between constraint solvers and the proof engine.

Another important step towards the deployment of CILL for real robotic planning problems is an efficient implementation that is not only capable of overcoming various fundamental problems in writing effective theorem provers for linear logic [5], but also provides ways in which existing decision procedures for a variety of constraint domains can be modularly incorporated into the system. Even though the formalization of the proof theory goes a long way towards enforcing modularity, there are still architectural issues to be resolved before a fully practical implementation is achieved.

Notwithstanding these issues, the set of possible applications of CILL for robotic planning problems is quite rich. Overall, Constrained Intuitionistic Linear Logic provides a uniform framework in which hybrid properties of a system as well as goals incorporating both discrete and continuous elements can be combined. We believe that this framework, complemented with physically meaningful formal semantics for a usable fragment as well as a variety of efficient decision procedures for relevant constraint domains promises to bridge the wide gap between discrete and continuous reasoning methods in a principled yet practical way.

V. ACKNOWLEDGMENTS

We thank Kaustuv Chaudhuri for his contributions and ideas on formalizing various aspects of CILL and Alfred A. Rizzi for his ideas and feedback on formulating application examples for CILL within robotics.

REFERENCES

[1] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proc. of the IEEE*, 88:971–984, 2000.

[2] M. Bozzano. *A Logic-Based Approach to Model Checking of Parameterized and Infinite-State Systems*. PhD thesis, U. of Genova, 2002.

[3] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*, 18(6):534–555, 1999.

[4] B.-Y. E. Chang, K. Chaudhuri, and F. Pfenning. A judgmental analysis of linear logic. Technical Report CMU-CS-03-131R, Carnegie Mellon University, December 2003.

[5] K. Chaudhuri and F. Pfenning. A focusing inverse method theorem prover for first-order linear logic. In *Proceed. of the 20th Int. Conf. on Automated Deduction*, Tallinn, Estonia, July 2005.

[6] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 2000.

[7] N. Davoren, J. M. Logics for hybrid systems. *Proc. of the IEEE*, 8(7):985–1010, 2000.

[8] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for mobile robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2032–2037, April 2005.

[9] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[10] G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[11] J.-Y. Girard. Linear logic: its syntax and semantics. In *Proc. of the workshop on Adv. in linear logic*, pages 1–42. Cambridge University Press, 1995.

[12] L. Jia and D. Walker. ILC: A foundation for automated reasoning about pointer programs. In P. Sestoft, editor, *Proc. of the 15th European Symp. on Programming Languages and Systems*, pages 131–145, Vienna, Austria, March 2006. Springer Verlag LNCS 3924.

[13] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. To be published in 2006.

[14] P. Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60, 1996.

[15] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. reprinted in McC90.

[16] D. C. Oppen. Complexity, convexity and combinations of theories. *Theoretical Computer Science*, 12(3):291–302, Nov. 1980.

[17] F. Pfenning. Structural cut elimination in linear logic. Technical Report CMU-CS-94-222, Cepartment of Computer Science, Carnegie Mellon University, December 1994.

[18] F. Pfenning. Lecture Notes on Linear Logic. Carnegie Mellon University, 2002.

[19] E. Rimon and D. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, October 1992.

[20] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[21] J. Slaney and S. Thiebaux. Blocks world revisited. *Artificial Intelligence*, 125(1-2):119–153, Jan. 2001.

[22] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das. The CLARAty architecture for robotic autonomy. In *IEEE Proc. of the Aerospace Conference*, volume 1, pages 121–132, Big Sky, MT, March 2001.