Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 125 (2005) 117–135
www.elsevier.com/locate/entcs

Resource Graphs and Countermodels in Resource Logics

Didier Galmiche and Daniel Méry [1]

LORIA UMR 7503 - Université Henri Poincaré
Campus Scientifique, BP 239
54506 Vandœuvre-les-Nancy, France

Abstract

In this abstract we emphasize the role of a semantic structure called resource graph in order to study the provability in some resource-sensitive logics, like the Bunched Implications Logic (BI) or the Non-commutative Logic (NL). Such a semantic structure is appropriate for capturing the particular interactions between different kinds of connectives (additives and multiplicatives in BI, commutatives and non-commutatives in NL) that occur during proof-search and is also well-suited for providing countermodels in case of non-provability. We illustrate the key points with a tableau method with labels and constraints for BI and then present tools, namely BILL and CheckBI, which are respectively dedicated to countermodel generation and verification in this logic.

Keywords: resources, proof-search, semantics, labels, countermodels.

1 Introduction

Over the past few years there has been an increasing amount of interest for resource-sensitive logical systems. The notion of *resource* is a basic one in many fields, including in computer science. The location, ownership, access to and, indeed, consumption of, resources are central concerns in the design of systems, such as networks, and in the design of programs, which access memory and manipulate data structures like pointers. Among so-called resource logics, we can mention Linear Logic (LL) [11] with its resource consumption interpretation, and Bunched Implications logic (BI) [15,16] with its

[1] Email: galmiche@loria.fr, dmery@loria.fr

doi:10.1016/j.entcs.2004.06.069

resource sharing interpretation but also order-aware (non-commutative) logic (NL) [1]. As specification logics, they can represent features as interaction, resource distribution and mobility, non-determinism, sequentiality or coordination of entities. For instance, BI has been recently used as an assertion language for mutable data structures [12] and in this context it is important to verify pre- or post-conditions expressed in this logic, mainly to discover non-theorems and if possible to provide explanation about this non-provability by generating readable and usable countermodels.

For the above mentioned resource logics, proof search is not trivial mainly because of the management of context splitting and bunches in the related sequent calculi. Moreover, the design of semantic-based methods is difficult because the semantics of such logics (like Grothendieck topological semantics for BI [16]), even if they are complete, are not always manageable in the context of proving or disproving formulae. Known methods, like tableaux or connections, dedicated to classical, intuitionistic or linear logics by using prefixes [13] cannot be easily extended to other resource logics. Therefore, in order to deal with the particular interactions occuring between connectives, for instance additive and multiplicative connectives in BI or commutative and non-commutative connectives in NL, our proposal is to start with a standard proof-search method (tableau or connection-based) and to define, for each logic, specific *labels* and (label) *constraints* that allow us to capture the interactions at a semantic level. It leads to the design of new calculi with labelled signed formulæ and constraints from which we define a new characterization of provability from standard notions, such as complementarity and closure conditions, extended with specific conditions about constraint satisfaction with respect to a particular set of constraints. This set is built during the proof-search process (tableau expansions or connection search) and can be easily represented as a graph, called *dependency graph*. It arises as the central syntactico-semantic structure from which the provability in some resource logics can be studied and allows us to generate countermodels, for instance, in Grothendieck topological semantics that is complete for BI. Another interesting point is to consider such a structure, with an appropriate valuation attached to some nodes, directly as a countermodel.

The relationships between semantics and syntax (labels and constraints) used to defined labelled calculi can be studied in both directions. For instance, in the case of BI without $\bot$, the labels and constraints directly reflect the elementary Kripke semantics of the logic [8] and thus the relationships between semantics and dependency graphs is clearly identified. In the case of BI (with $\bot$), the labels and constraints do not reflect the initial Grothendieck topological semantics, but considering the dependency or *resource graph*

as the right representation of countermodels we can define a new simple resource semantics which is complete for BI [9] and which is a direct reflection of the labelled calculus. A key point to mention is that these notions of labels, constraints and resource graphs are not limited to tableaux methods but can also be considered from the perspective of connection-based proof-methods. It emphasizes that the semantic knowledge required to analyze provability is mainly covered by the resource graphs built in parallel with the standard proof-search methods. As said before, the approach is not only applicable to BI but also to Non-commutative logic (NL) for which we do not initially have a useful resource semantics but only a bunched calculus not well adapted to proof-search. In the case of NL, we are able to define a connection-based method using appropriate labels and resource graphs that capture its particular semantics. Knowing that, in linear logics, connection methods and proof nets (a standard semantic structure) are closely related [6], we then deduce an algorithm that builds proof nets. In case of non-provability, the partial proof net under construction and the resource graph both provide some explanations about the non-validity [10]. Further work will be devoted in this context to build countermodels in the corresponding phase semantics and also to define a new semantics in which the resource graph can be seen as a countermodel.

In section 2 we focus on the notions of resource and dependency graph also called, under some conditions, resource graph. This new semantic structure is central in the design of proof-search methods for resource logics like BI and allows us to generate countermodels in order to analyze the non-validity. In section 3, we emphasize the relationships between a resource logic, its semantics or its particular sequent calculus and the definition and construction of specific dependency graphs from which provability can be discussed. In order to illustrate the main ideas and results about labels, constraints and resource graphs, we consider here the BI logic with an approach based on tableaux, but similar ideas can be applied to different resource logics such as MILL or NL and to different proof-search methods such as connections or natural deduction. In section 4, we describe the BILL system, an automated theorem prover for propositional BI, that implements the previous results and builds proofs or countermodels in this logical fragment. We also consider the possibility of verifying models or countermodels through the description of a model-checker, called CheckBI. Further work will be devoted to the improvement of proof-search in resource logics by combining theorem-proving and model-checking approaches. For instance, we could improve the BILL and CheckBI systems and study how to combine their use to prove or disprove formulæ more efficiently. Moreover, starting from these theoretical and practical results, we expect to propose similar methods and tools with countermodel

generation for separation logics [12,17] and spatial logics [2,3].

2 Resources and Resource Graphs

Let us formalize the notion of resource in an elementary way that is sufficient for our purpose. We start with a set R of resources with some properties that appear as characteristic: the existence of an *initial resource* or unit, denoted 1; the existence of a composition operator $\cdot$ that combines two resources x and y into a new one denoted $x \cdot y$; the existence of an operator $\leq$ which compares two resources x and y. At this level, our notion of resource is elementary because it does not consider the location or the ownership of a resource.

We may state additional conditions on the comparison of resources, for instance, reflexivity ($x \leq x$) and/or transitivity ($x \leq y$ and $y \leq z$ imply $x \leq z$). We may also impose particular conditions on resource-composition such as associativity ($x \cdot (y \cdot z) = (x \cdot y) \cdot z$), commutativity ($x \cdot y = y \cdot x$), identity w.r.t. 1 ($1 = 1 \cdot x = x$) or compatibility with $\leq$ ($x \leq y$ implies $x \cdot z \leq y \cdot z$). The compatibility of resource-composition w.r.t. $\leq$ is a natural property in many resource-settings. For example, if we consider the resources x and y as files and we interpret the comparison $x \leq y$ of two files as "x has shorter length than y" then, taking the composition $x \cdot y$ as the concatenation of x and y, the compatibility condition means that appending any file z to x always results in a file that has shorter length than the one obtained by appending z to y, provided that the file x is initially shorter than the file y.

Let us consider now what we call a *resource graph*. It is a directed graph $G(N, E)$ with N a set of nodes and E a set of edges between nodes that satisfies some specific properties. The nodes of the graphs are *labels*. The labelling language consists of the following symbols: a unit symbol 1, a binary function symbol $\circ$, a binary relation symbol $\leq$, a countable set of constants $c_1, c_2, \ldots$ *Labels* are inductively defined from the unit 1 and the constants as expressions of the form $x \circ y$ in which x and y are labels. *Atomic labels* are labels which do not contain any $\circ$, while *compound labels* contain at least one $\circ$. A *sublabel* of a label x is a subterm of x. If we take $\circ$ to be monoidal then labels can be interpreted as multi-sets, 1 being the empty-set and $\circ$ being multi-set union. We can therefore omit the symbol $\circ$ when writing labels, for instance, $c_2c_2c_5$ represents the multi-set $\{c_2, c_2, c_5\}$ and the composition of the labels $c_1c_3c_4$ and c_2c_5 is the label $c_1c_2c_3c_4c_5$. Moreover, two labels are equivalent if they contain the same occurrences of constants. For instance, $c_1c_2c_2$, $c_2c_1c_2$, $1c_2c_1c_2$, $11c_1c_2c_2$ denote the same label (1 is not a constant). The notion of sublabel simply corresponds to the notion of sub-multi-set : y is a sublabel of x (notation $y \subseteq x$) if the multi-set denoted by y is included in

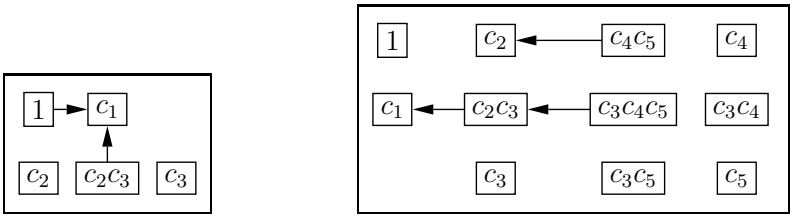


Figure 1. Examples of Resource Graphs

the multi-set denoted by x. For instance, c_1c_3 is a sublabel of $c_1c_2c_3c_4$. Then, $\mathcal{P}(x)$ represents the set of the sublabels of x.

Let $G(N, E)$ a directed graph, the nodes of N being labelled as explained above. We denote $x \to y$ the oriented edge between the nodes x and y and $x \to^* y$ a path from x to y.

From now on, we can consider that labels represent resources and edges represent links between resources. To be a resource graph, a directed graph $G(N, E)$ must satisfy the following conditions:

(i) $(\forall x \in N)(\mathcal{P}(x) \subseteq N)$ (closure under sublabels);

(ii) $yz \in N$ and $x \to y \in E$ imply $xz \in N$ and $xz \to yz \in E$ (partial compatibility).

The first condition means that if a label is in the graph, its sublabels also are. The second condition corresponds to a weak form of compatibility of the composition of resources w.r.t. the preordering. Figure 1 presents examples of resource graphs.

A resource graph is a graphical representation of a set of resources which can be composed and which verifies some particular conditions. In this abstract, we focus on the central role played by the resource graph as a semantic structure for proving and disproving formulæ in resource-sensitive logics which provides explanations of non-provability through the generation of countermodels.

3 Proofs, Resource Graphs and Countermodels

Given a resource-aware logic, having (bunched or not) sequent calculi and related semantics, the design of proof-search methods is not trivial because of the management of formulæ as resources (context splitting, interactions). We aim to illustrate the relationships between the way resources are managed, in bunched calculi or in resource semantics, and the definition of specific resource graphs from which proving or disproving can be studied.

In order to illustrate the main ideas and results about labels, constraints

$$\frac{}{\phi \vdash \phi}\, ax \qquad \frac{\Gamma \vdash \phi}{\Delta \vdash \phi}\, \Delta \equiv \Gamma \qquad \frac{\Gamma(\Delta) \vdash \phi}{\Gamma(\Delta;\Delta') \vdash \phi}\, w \qquad \frac{\Gamma(\Delta;\Delta) \vdash \phi}{\Gamma(\Delta) \vdash \phi}\, c \qquad \frac{\Delta \vdash \phi \quad \Gamma(\phi) \vdash \psi}{\Gamma(\Delta) \vdash \psi}\, cut$$

$$\frac{}{\Gamma(\bot) \vdash \phi}\, \bot_L \qquad \frac{\Gamma(\emptyset_m) \vdash \phi}{\Gamma(\mathrm{I}) \vdash \phi}\, \mathrm{I}_L \qquad \frac{}{\emptyset_m \vdash \mathrm{I}}\, \mathrm{I}_R \qquad \frac{\Gamma(\emptyset_a) \vdash \phi}{\Gamma(\top) \vdash \phi}\, \top_L \qquad \frac{}{\emptyset_a \vdash \top}\, \top_R$$

$$\frac{\Gamma(\phi,\psi) \vdash \chi}{\Gamma(\phi * \psi) \vdash \chi}\, *_L \qquad \frac{\Gamma \vdash \phi \quad \Delta \vdash \psi}{\Gamma,\Delta \vdash \phi * \psi}\, *_R \qquad \frac{\Delta \vdash \phi \quad \Gamma(\psi,\Delta') \vdash \chi}{\Gamma(\Delta,\phi \mathrel{-\!\!*} \psi,\Delta') \vdash \chi}\, \mathrel{-\!\!*}_L \qquad \frac{\Gamma,\phi \vdash \psi}{\Gamma \vdash \phi \mathrel{-\!\!*} \psi}\, \mathrel{-\!\!*}_R$$

$$\frac{\Gamma(\phi;\psi) \vdash \chi}{\Gamma(\phi \wedge \psi) \vdash \chi}\, \wedge_L \qquad \frac{\Gamma \vdash \phi \quad \Delta \vdash \psi}{\Gamma;\Delta \vdash \phi \wedge \psi}\, \wedge_R \qquad \frac{\Delta \vdash \phi \quad \Gamma(\psi;\Delta') \vdash \chi}{\Gamma(\Delta;\phi \to \psi;\Delta') \vdash \chi}\, \to_L$$

$$\frac{\Gamma;\phi \vdash \psi}{\Gamma \vdash \phi \to \psi}\, \to_R \qquad \frac{\Gamma(\phi) \vdash \chi \quad \Gamma(\psi) \vdash \chi}{\Gamma(\phi \vee \psi) \vdash \chi}\, \vee_L \qquad \frac{\Gamma \vdash \phi_{i\,(i=1,2)}}{\Gamma \vdash \phi_1 \vee \phi_2}\, \vee_{Ri}$$

Figure 2. The LBI Sequent Calculus

and dependency graphs, we consider the BI logic with a tableau-based proof-method, but it is important to notice that our approach can also be applied to other resource logics like MILL or NL and to other proof methods.

3.1 Resources and BI logic

The development of a mathematical theory of resource is one of the objectives of the programme of study of the logic of bunched implications (BI) [15,16]. The basic idea is to model directly the observed properties of resources and then to give a logical axiomatization. This logic provides a logical analysis of a basic notion of resource, quite different from linear logic's "number-of-uses" reading, which has proved rich enough to provide "pointer logic" semantics for programs which manipulate mutable data structures [12,14]. In this context, proof-search methods are necessary and the generation of countermodels in order to provide explanations of non-provability is very important.

The propositional language of BI consists of: a multiplicative unit I, the multiplicative connectives $*$, $\mathrel{-\!\!*}$, the additive units $\top$, $\bot$, the additive connectives $\wedge$, $\to$, $\vee$, a countable set $L = \mathrm{p}, \mathrm{q}, \ldots$ of propositional letters. $\mathcal{P}(L)$, the collection of BI propositions over L, is given by the following inductive definition: $\phi ::= p \mid \mathrm{I} \mid \phi * \phi \mid \phi \mathrel{-\!\!*} \phi \mid \top \mid \bot \mid \phi \wedge \phi \mid \phi \to \phi \mid \phi \vee \phi$. The additive connectives correspond to those of intuitionistic logic (IL) whereas the multiplicative connectives correspond to those of multiplicative intuitionistic linear logic (MILL).

The antecedents of logical consequences are structured as bunches which are trees representing the two ways (";" and ";") of combining BI formulæ to respectively display additive or multiplicative behavior. More formally, bunches

are given by the grammar: $\Gamma ::= \phi \mid \emptyset_a \mid \Gamma\,;\Gamma \mid \emptyset_m \mid \Gamma\,,\Gamma$. Equivalence of bunches, $\equiv$, is given by commutative monoid equations for "," and ";", whose units are $\emptyset_m$ and $\emptyset_a$ respectively, together with the evident substitution congruence for subbunches. $\Gamma(\Delta)$ denotes a subbunch Δ of Γ. Judgements are expressions of the form $\Gamma \vdash \phi$, where Γ is a "bunch" and ϕ is a proposition.

The LBI sequent calculus is given on Figure 2. We say that a proposition ϕ is a theorem if and only if $\emptyset_m \vdash \phi$ is provable in LBI.

BI has a natural Kripke-style semantics (interpretation of formulæ) which combines Kripke's semantics for IL and Urquhart's semantics for MILL [15]. This semantics deals with possible worlds, arranged as a commutative monoid and justified in terms of "pieces of information". It provides a way to read the formulæ as propositions that are true or false relative to a given world. BI's Kripke semantics may be adapted to take $\bot$ into account by moving from presheaves (elementary semantics) to sheaves on a topological space, namely, Grothendieck topological semantics [16]. Such a semantics considers an inconsistent world at which $\bot$ is forced together with the standard so-called indecomposable treatment of the disjunction, *i.e.*, $m \models \phi \vee \psi$ if and only if $m \models \phi$ or $m \models \psi$.

3.2 Proof-search and Resource Graphs

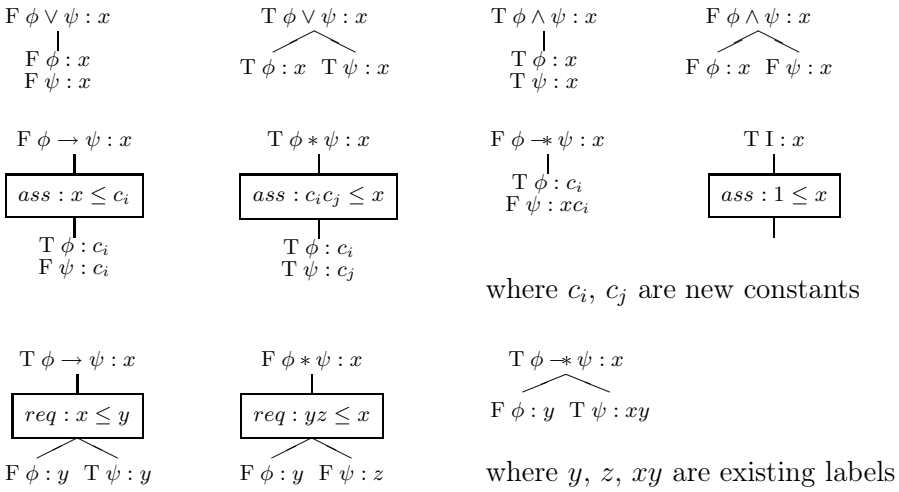


Figure 3. The Tableau Expansion Rules

Having in mind the completeness results for BI's semantics w.r.t. the LBI

sequent calculus, we aim to study the proof-theoretical foundations of (propositional) BI in order to propose proof-search methods that build proofs or countermodels. For that, the challenge is to capture the *interactions* between the various kinds of connectives, interactions which are reflected in bunched sequent calculi by means of structural rules that lead to complex and subtle transformations of the bunches. Our idea is to capture the interactions at a semantic level using labels and constraints in the spirit of labelled deductive systems [4]. A key step in our semantic analysis is the use of so-called *dependency graphs* that are in fact particular resource graphs.

Let us illustrate these points with the BI logic. We define labels and sublabels as in section 2. *Label constraints* are expressions of the form $x \leq y$, where x and y are labels. We deal with *partially defined* labelling algebras, obtained from sets of labels and constraints by reflexive, transitive and partial compatible closure. We note $\overline{K}$ the closure of K, where K is a set of labels and constraints and we say that a constraint $x \leq y$ is a consequence of (or satisfied by) K if $x \leq y$ is in $\overline{K}$.

Having defined such labels and constraints, we can define new labelled calculi (sequent, tableaux or connections) such that, in parallel with the standard proof-search process, one generates a resource graph (set of particular constraints), from which we can analyze the provability. We illustrate the main points with a tableau method that is well-adapted for a direct generation of countermodels. Compared to the standard method we consider *signed formulæ* $\mathrm{Sg}\,\phi : l$ with Sg $(\in \{F, T\})$ being the sign of the formula ϕ and l its label. Then we define a labelled calculus that consists of the expansion rules of Figure 3. We observe that we have $\pi\alpha$ rules that introduce constraints called *assertions* (including $F{-\!\!*}$ for which the assertion $c_i \leq c_i$ is implicit) and $\pi\beta$ rules that introduce constraints called *requirements*. The set of all the assertions occuring in a branch $\mathcal{B}$ is denoted $Ass(\mathcal{B})$ while the set of all its requirements is denoted $Req(\mathcal{B})$.

Building a labelled tableau for an initial signed formula $\mathrm{F}\,\phi : 1$, by application of the expansion rules of Figure 3, the key problem is to define branch closure conditions such that, either the tableau is closed and then ϕ is valid, or there exists an open branch and then ϕ is not valid [5]. Moreover, in the latter case, we aim to build a countermodel for ϕ from an open branch. To this end, given a tableau branch $\mathcal{B}$ we define its associated resource graph $DG(\mathcal{B})$ as the directed graph $G(N, E)$ such that there is a node labelled with x in N if and only if there is a label x in $\overline{Ass}(\mathcal{B})$ and there is an edge $x \rightarrow y$ in E if and only if there is an assertion $x \leq y$ in $\overline{Ass}(\mathcal{B})$. Let us note that $\pi\alpha$ rules create new (atomic) labels while $\pi\beta$ are supposed to reuse the ones that already exist in the resource graph associated to a branch. The example of

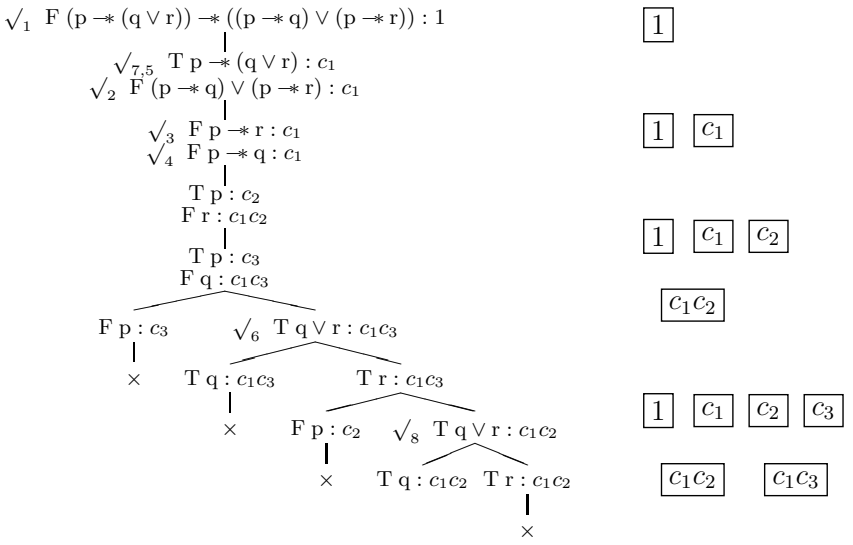


Figure 4. Tableau for $(p \multimap (q \vee r)) \multimap ((p \multimap q) \vee (p \multimap r))$

Figure 4 illustrates the parallel construction of a tableau and a resource graph related to the set of assertions.

In this context we say that two signed formulæ $T\ \phi : x$, $F\ \phi : y$ are *complementary* in a branch $\mathcal{B}$ if and only if $x \leq y \in \overline{Ass}(\mathcal{B})$, *i.e.*, there is a path from x to y in $DG(\mathcal{B})$. This definition can be seen as an extension of the standard notion of complementarity in standard labelled deductive systems. We also need to take care of the units, with a particular difficulty on $\bot$, the unit of the disjunction $\vee$. For that, we introduce the notion of an inconsistent label in a tableau branch. We say that a label x is *inconsistent in* $\mathcal{B}$ if there exists a label y such that $y \leq x \in \overline{Ass}(\mathcal{B})$ and a sublabel z of y, such that $T \bot : z$ occurs in $\mathcal{B}$. To complete the closure conditions, we impose that each requirement $x \leq y$ occuring in a branch $\mathcal{B}$ of a tableau t should be a consequence of the assertions occuring above it in $\mathcal{B}$ (closer to the root of t). More details are given in [9]. To summarize, a tableau t is *closed* iff each branch $\mathcal{B}$ in t satisfies the following conditions:

(i)
1. $\mathcal{B}$ contains two complementary formulæ $T\ \phi : x$ and $F\ \phi : y$, or
2. $\mathcal{B}$ contains a formula $F \top : x$, or
3. $\mathcal{B}$ contains a formula $F\ I : x$ with $1 \leq x \in \overline{Ass}(\mathcal{B})$, or
4. $\mathcal{B}$ contains a formula $F\ \phi : x$ with x inconsistent in $\mathcal{B}$;

(ii) $\forall\, x \leq y \in Req(\mathcal{B})$, $x \leq y \in \overline{Ass}(\mathcal{B})$.

If we suppress the condition (i).4 in the previous definition, we obtain closure conditions that fit well with BI without $\bot$ and its elementary Kripke semantics. We can also show that, with this additional condition, we can cope with BI and its Grothendieck topological semantics.

Coming back to our example of Figure 4, we observe that the tableau has four branches (marked with a cross $\times$) which are closed because of complementary formulæ, but the fourth branch remains open. Thus, we can conclude that the BI formula is not provable. In the next subsection, we explain how to generate a countermodel from such an open branch and the associated resource graph.

3.3 Completeness and Countermodel Generation

First, we can show that the resource tableau method is sound with respect to the Grothendieck topological semantics.

Theorem 3.1 (soundness) *Let ϕ be a proposition of BI, if there exists a closed tableau $\mathcal{T}$ for ϕ then ϕ is valid.*

Details of the proof are given in [9]. It is not a simple extension of the proof of [8] because, with $\bot$, we have to deal with Grothendieck topological semantics.

In order to prove the completeness of the method, we have to define how to build a countermodel for ϕ from an open branch in a labelled tableau and its associated resource graph, which represents the reflexive, transitive and partial compatible closure of the assertions. Therefore, if a formula ϕ happens to be unprovable, we should have enough information in the resource graph to extract a countermodel for ϕ. The idea underlying the countermodel construction is to regard the resource graph itself as the desired countermodel, thereby considering it as a central semantic structure. For that, we consider the nodes (labels) of the graph as the elements of a monoid whose multiplication is given by the composition of labels.

The key point is that, since the closure operator induces a *partially defined* labelling algebra, the resource graph only deals with those pieces of information (resources) that are relevant for deciding provability. Therefore, the monoidal product should be completed with suitable values for those compositions which are undefined. The problem of undefinedness is then solved by the introduction of a particular element, denoted π, to which all undefined compositions are mapped.

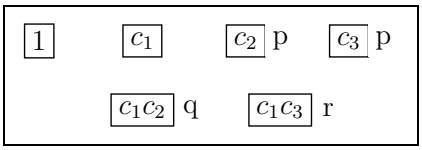
More precisely, in order to transform the resource graph $G(N, E)$ into a resource monoid $\langle \mathrm{R}, \cdot, 1, \leq \rangle$, we add a special node π to N, *i.e.*, $\mathrm{R} \stackrel{\text{def}}{=} N \cup \{\pi\}$. Then, the monoidal product $\cdot$ is given by $x \cdot y \stackrel{\text{def}}{=} xy$ if $xy \in N$ and $x \cdot y \stackrel{\text{def}}{=} \pi$

otherwise. Notice that any composition with something undefined is itself undefined. The preordering relation is given by the arrows of the graph as follows: $x \leq y$ if and only if $x \rightarrow^* y$ or $y = \pi$, π being the greatest element. Finally, the forcing relation simply reflects the signed formulæ of the open branch.

Theorem 3.2 (completeness) *Let ϕ be a proposition of BI, if ϕ is valid then there exists a closed tableau $\mathcal{T}$ for ϕ.*

The proof is detailed in [9] in which we also deduce the decidability and the finite model property for propositional BI as main results.

Returning to our example of Figure 4, we build a countermodel from the open branch $\mathcal{B}$ by considering the signed formulæ of $\mathcal{B}$ of the form $\mathrm{T}\, At : x$ where At is an atom. The set of labels attached to At is considered as its valuation and we complete the resource graph, at each node or label, with the corresponding atoms. Thus, we have the valuation v such that $v(\mathrm{p}) = \{c_2, c_3\}$, $v(\mathrm{q}) = \{c_1c_2\}$ and $v(\mathrm{r}) = \{c_1c_3\}$ and then represent the following resource graph:



We show that the node labelled with 1 does not force the formula ϕ, *i.e.*, $G, 1 \not\models_v \phi$. Indeed, both (i) $c_1 \models \mathrm{p} \mathrel{-\!\!*} (\mathrm{q} \vee \mathrm{r})$ and (ii) $c_1 \not\models (\mathrm{p} \mathrel{-\!\!*} \mathrm{q}) \vee (\mathrm{p} \mathrel{-\!\!*} \mathrm{r})$ hold; therefore, $1 \not\models (\mathrm{p} \mathrel{-\!\!*} (\mathrm{q} \vee \mathrm{r})) \mathrel{-\!\!*} ((\mathrm{p} \mathrel{-\!\!*} \mathrm{q}) \vee (\mathrm{p} \mathrel{-\!\!*} \mathrm{r}))$ follows from $1c_1 = c_1$ and the definition of $\mathrel{-\!\!*}$. Let us show now (i) and (ii). For (i), we observe that $c_1c_2 \models \mathrm{q} \vee \mathrm{r}$ since $c_1c_2 \models \mathrm{q}$ and that $c_1c_3 \models \mathrm{q} \vee \mathrm{r}$ since $c_1c_3 \models \mathrm{r}$. The nodes c_2 and c_3 that force p can be combined with c_1 to provide c_1c_2 and c_1c_3. As we have $c_1c_2 \models \mathrm{q} \vee \mathrm{r}$ and $c_1c_3 \models \mathrm{q} \vee \mathrm{r}$, by definition of $\mathrel{-\!\!*}$, we deduce $c_1 \models \mathrm{p} \mathrel{-\!\!*} (\mathrm{q} \vee \mathrm{r})$. For (ii), we have $c_1 \not\models \mathrm{p} \mathrel{-\!\!*} \mathrm{q}$ since $c_3 \models \mathrm{p}$ and $c_1c_3 \not\models \mathrm{q}$. We also have $c_1 \not\models \mathrm{p} \mathrel{-\!\!*} \mathrm{r}$ since $c_2 \models \mathrm{p}$ and $c_1c_2 \not\models \mathrm{r}$. Thus, we have neither $c_1 \models \mathrm{p} \mathrel{-\!\!*} \mathrm{q}$, nor $c_1 \models \mathrm{p} \mathrel{-\!\!*} \mathrm{r}$, *i.e.*, $c_1 \not\models (\mathrm{p} \mathrel{-\!\!*} \mathrm{q}) \vee (\mathrm{p} \mathrel{-\!\!*} \mathrm{r})$.

The previous results and examples illustrate the central role played by the resource graphs for the generation of countermodels. Thus, we can extract, from the resource graph, a countermodel in the related semantics, *i.e.*, in the Kripke elementary semantics for BI without $\bot$ (our example) but also in the Grothendieck topological semantics for BI with $\bot$ [9]. As said before, for a resource logic like BI, we can relate a resource graph with a given complete semantics like Grothendieck topological semantics. But an interesting question arises: is it possible to deduce a new resource semantics from a deeper analysis of a given resource graph ? In the case of BI, the answer is yes. By consid-

ering resource graphs directly as countermodels, we have recently proposed a new semantics based on partially defined monoids (in which the monoidal operation is partial) that reflects the natural treatment of $\perp$ in a resource graph generated by our approach [9]. The existence of such a semantics that generalizes the models of BI pointer logic [12] was an open question and it is clear that the resource graph is the central notion allowing to give a positive answer.

3.4 A General Methodology for Resource Logics

We have extended the standard tableau method [4] with labels and constraints related to some resource semantics. The resulting method is based on two parallel processes: a syntactic decomposition of the formula to be proved together with a semantic construction of a resource graph from which provability can be determined. Thus, we extend standard conditions (here closure of branches) with label management w.r.t. a resource graph and from this new semantic structure one can prove or disprove formulæ and generate proofs or countermodels. A similar approach based on labels, constraints and resource graphs has been used in order to define a connection-based method for propositional BI [7]. It emphasizes the fact that we propose a general methodology based on the construction of resource graphs associated to standard proof-methods such as tableaux or connections.

But a question arises: is this methodology restricted to BI logic ? A first answer comes from the fact that BI is conservative over intuitionistic logic (IL) and multiplicative intuitionistic linear logic (MILL) [15,16], which implies that our new proof-search methods can be restricted to both logics. It provides a new method for IL in which prefixes and unification [13] are replaced by labels and constraint-solving. Moreover, it is well adapted to the generation of countermodels. Further works will be devoted to deeper comparisons from the perspectives of proof-search efficiency and countermodel construction. In addition, we obtain the first tableau (and connection) method for MILL, which well illustrates the power of resource graphs for such resource logics. Knowing the relationships between connection-based characterizations and proof nets in linear logics [6], our results also lead to a new algorithm for automated construction of MILL proof nets. Moreover, from the semantic point of view, the impact of these results will be analyzed and compared with previous proposals for the analysis of models and countermodels.

The previously mentioned resource logics are directly related to BI but we can also consider other resource logics like Multiplicative Non-commutative Logic (MNL), which is a conservative extension of both commutative (MLL) and non-commutative or cyclic (MCyLL) linear logic [1]. Specific labels, con-

straints and resource graphs can be defined in order to capture the interactions between commutative and non-commutative connectives (and thus its phase semantics) in this logic. They give rise to the definition of a connection-based characterization of provability in MNL [10] and a related proof-search method. Finally, we aim to extend our results to proof-search and verification in separation logics [12,17] and spatial logics [2,3].

4 Countermodel Generation and Verification

Let us first consider the proof-search approach implemented in the BILL system. In order to illustrate its principles we consider our example formula ϕ defined as $\phi \equiv (p \multimap (q \vee r)) \multimap ((p \multimap q) \vee (p \multimap r))$ and show that is not valid by generating a countermodel as a resource graph. Then, we complete the presentation with the model-checking approach and its implementation, the CheckBI system, that verifies that the resource graph built by the BILL prover really is a countermodel for the formula ϕ.

4.1 BILL and Countermodel Generation

BILL is a prover for propositional BI (http://www.loria.fr/~dmery/BILL), written in CAML, that is able to decide whether a BI formula is provable or not and thus to build a countermodel under the form of a particular graph representation, that is a resource graph. In its current version, BILL can export the generated countermodels as GDL (Graph Description Language) files, GDL being a variant of XML adapted to graph descriptions. Thus, starting with a non-provable BI formula ϕ, a user can obtain, in a GDL file, a resource graph that is a countermodel for ϕ.

Let us describe the BILL prover and its main characteristics. It can be seen as an interpreter with simple commands. Its main command is `check <formula>`, that allows to decide the validity of `<formula>`. This `<formula>` parameter is written with the following syntax: the additive conjunctive unit is `1`; the additive disjunctive unit is `0`; the multiplicative unit is `I`; the additive connectives are: `^` (and), `v` (or), `->` (implication); the multiplicative connectives are `*` (star), `-*` (magicwand); the propositional variables are alphabetic characters, (`p`,`q`,...), except `I`, reserved for the multiplicative unit, and `v`, reserved for the additive disjunction.

Figure 5 illustrates the use of the BILL system. The `bill` command starts a BILL session and the `BILL>` prompt indicates that the user can write commands. With the `help` command we obtain a brief summary of the syntax of the formulæ and of the available commands. Then, with the command `check`, the user asks whether the formula $(p \multimap (q \vee r)) \multimap ((p \multimap q) \vee (p \multimap r))$

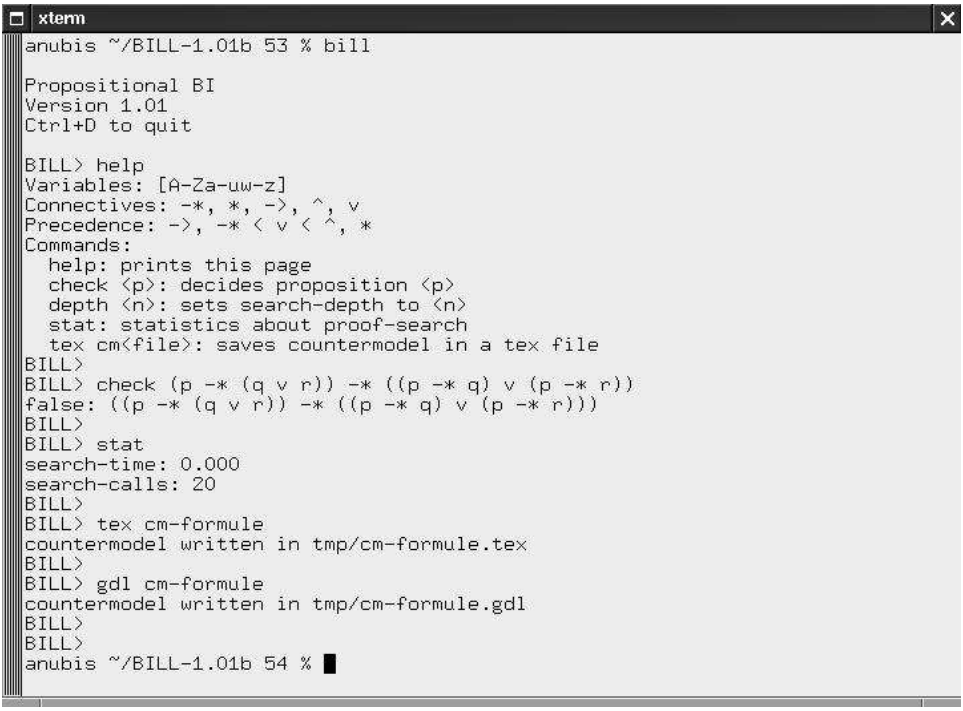


Figure 5. Example of a BILL Session

given as a parameter is valid. Then, BILL replies that it is not valid and the `stat` command displays the time and the number of recursive calls used to conclude.

The command `stat` gives informations about proof-search, for instance the time (in seconds) that is needed to decide the formula and the number of recursive calls of the proof-search loop. In case of non-validity, BILL can generate a countermodel with the commands `tex cm-<fichier>` and `gdl cm-<fich- ier>`. The first one generates a LaTeX file that describes and explain the semantic structure of the countermodel. The second one generates a countermodel as a resource graph with the GDL format. Such a graph can be exploited by graph manipulation tools such as `aiSee` ou `xvcg`.

As the formula is not valid, we aim to generate a countermodel both as a LaTeX file ans as a GDL resource graph. We do so using the commands `tex cm-formule` and `gdl cm-formule` that respectively provide the files `cm-formule.tex` and `cm-formule.gdl`. Figure 6 shows what we obtain after the treatment of `cm-formule.tex` by LaTeX. The document contains the resource graph that is a countermodel with a list of worlds, and for each world, the lists of its immediate successors. We can observe that BILL has

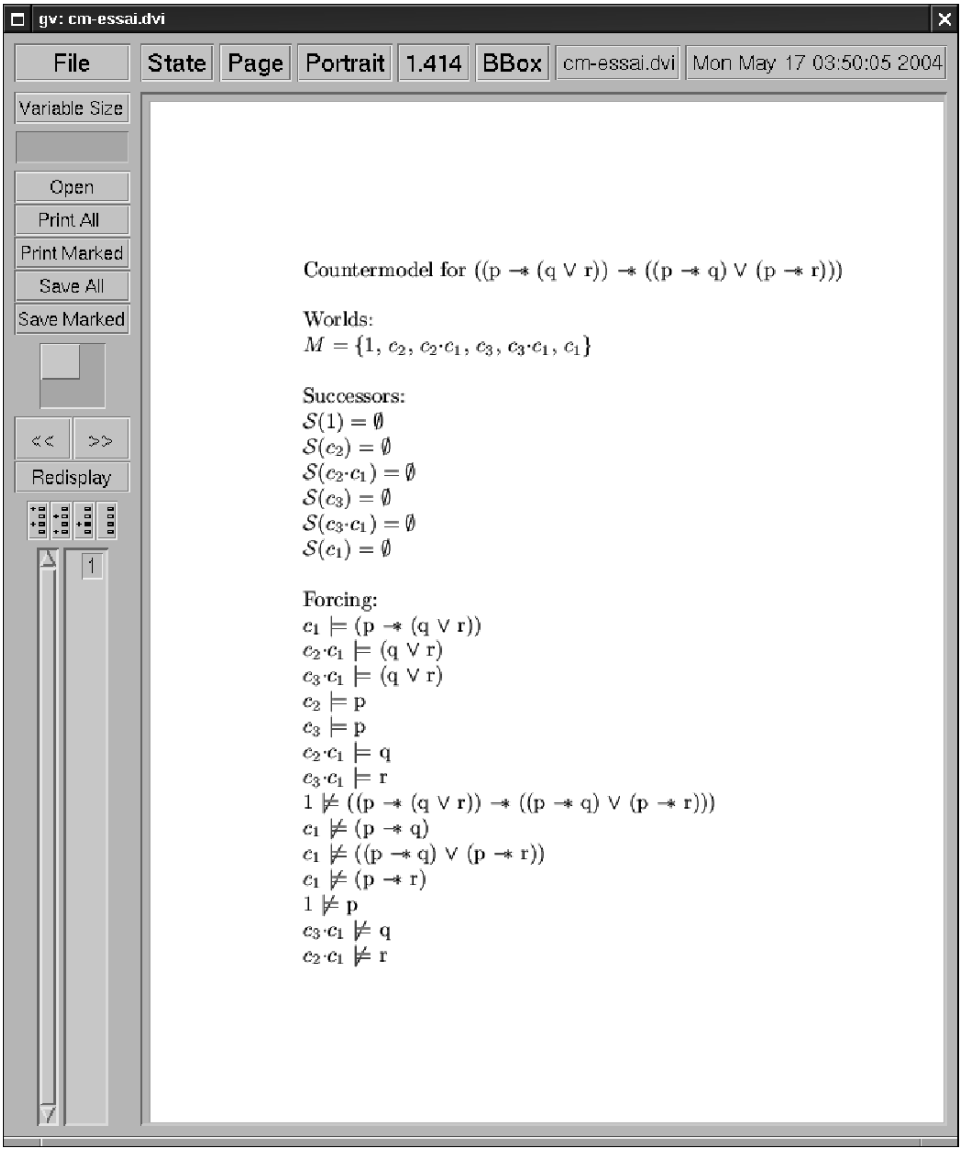


Figure 6. A Countermodel in a Latex file

generated the resource graph described in section 3. Moreover the document provides the explanations of the non-validity of the formula by describing, for each subformula, the worlds for which it is verified or falsified. As one can notice, the proof given in section 3 is recovered.

4.2 CheckBI and Countermodel Verification

Another tool, called CheckBI and written in Java, implements a dual functionality: verifying whether a resource graph is a countermodel for a given BI formula under some valuation of atomic formulæ over the nodes of the graph. More precisely, given a resource graph $G(N, E)$, a valuation v and a formula ϕ, CheckBI verifies if the formula ϕ is true for the graph G under the valuation v, *i.e.*, if $G, 1 \models_v \phi$ holds true. We see this tool as a first step towards studying the combination of the model-checking and theorem proving approaches in BI, *i.e.*, how to combine proof-search and countermodel-search in order to achieve efficient decision procedures including countermodel generation facilities. The CheckBI command line has the following syntax: `checkBI <graph> <formula> [<valuation>]`. The `<graph>` parameter corresponds to a file containing the GDL specification of the graph. The command verifies that this specification indeed corresponds to a resource graph, more precisely, a graph that respects the conditions described in section 2. The `<formula>` parameter is a file that contains a BI formula written with the syntax used in BILL. Finally, the `<valuation>` parameter describes a particular distribution of the atoms occuring in the formula `<formula>` over the nodes of the graph. The distribution is specified as a list of pairs (node, list of atoms forced at that node). This parameter is optional and if it is not present, it means that, for each node, there is no atom forced at that node.

We illustrate the use of this tool with the countermodel generated by BILL for our example. If this resource graph really is a countermodel, the CheckBI tool should reply that the graph falsifies the formula. Figure 7 shows the contents of the `cm-formule.gdl` file, that is the specification in GDL of the resource graph. We observe that it is the graph described in section 2. The `formule.txt` file contains the formula $(\mathrm{p} \multimap (\mathrm{q} \vee \mathrm{r})) \multimap ((\mathrm{p} \multimap \mathrm{q}) \vee (\mathrm{p} \multimap \mathrm{r}))$ and the `affec.aff` file describes the valuation $v(\mathrm{p}) = \{c_2, c_3\}$, $v(\mathrm{q}) = \{c_1 c_2\}$ and $v(\mathrm{r}) = \{c_1 c_3\}$, already mentioned in section 3. The `checkBI cm-formule.gdl formule.txt affec.aff` command provides the result that was expected, as we can check it in Figure 7.

5 Conclusion and Perspectives

The aim here is to focus on a semantic structure, called resource graph, which is well adapted to the definition of decision procedures that generate countermodels for various propositional resource logics. This graph structure, with an additional valuation of atomic formulæ attached to its nodes, leads to a nice graphical representation of a countermodel, thus avoiding its explicit representation in some resource semantics that are difficult to be dealt with, for

Figure 7. An Example of GDL Countermodel Verification

instance, topological semantics. Such a resource graph arises from the definition of calculi including labels and label constraints that allow to capture the semantic interactions between connectives. This approach is well adapted to the treatment of "mixed" resource logics in which connectives of different kinds cohabit. As BI is used as an assertion logic for mutable data structures [12] and is the logical kernel of so-called separation logics, our results and their implementation in the BILL system are important to support the development of correct programs with pointers or the verification of properties for semi-structured data. In a practical perspective, the graphical representation of countermodels provides helpful and readable information for some kind of failure analysis that we aim to develop in further work.

Another point to be studied is the combination of the theorem-proving and model-checking approaches in order to improve the proof-search process for our resource logics. It involves studying how our tools, BILL and CheckBI, can be mutually used to devise efficient proof and disproof tactics and also how they can be extended to deal, for instance, with pointer logic and with various fragments of separation logics. The relationships between resource graphs and countermodels in new resource semantics will also be more deeply explored.

References

[1] M. Abrusci and P. Ruet. Non-commutative logic I : the multiplicative fragment. *Annals of Pure and Applied Logic*, 101:29–64, 2000.

[2] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). In *4th Int. Symposium on Theoretical Aspects of Computer Software, TACS 2001, LNCS 2215*, pages 1–37, Sendai, Japan, October 2001.

[3] L. Cardelli, P. Gardner, and G. Ghelli. A spatial logic for querying graphs. In *Int. Conference on Automata, Langages and Programming, ICALP'02, LNCS 2380*, pages 597–610, 2002.

[4] M. D'Agostino and D.M. Gabbay. A Generalization of Analytic Deduction via Labelled Deductive Systems. Part I: Basic substructural logics. *Journal of Automated Reasoning*, 13:243–281, 1994.

[5] M. Fitting. *First-Order Logic and Automated Theorem Proving.* Texts and Monographs in Computer Science. Springer Verlag, 1990.

[6] D. Galmiche. Connection Methods in Linear Logic and Proof nets Construction. *Theoretical Computer Science*, 232(1-2):231–272, 2000.

[7] D. Galmiche and D. Méry. Connection-based proof search in propositional BI logic. In *18th Int. Conference on Automated Deduction, CADE-18, LNAI 2392*, pages 111–128, 2002. Copenhagen, Danemark.

[8] D. Galmiche and D. Méry. Semantic labelled tableaux for propositional BI without bottom. *Journal of Logic and Computation*, 13(5):707–753, 2003.

[9] D. Galmiche, D. Méry, and D. Pym. Resource Tableaux (extended abstract). In *16th Int. Workshop on Computer Science Logic, CSL 2002, LNCS 2471*, pages 183–199, September 2002. Edinburgh, Scotland.

[10] D. Galmiche and J.M. Notin. Connection-based Proof Construction in Non-commutative Logic. In *10th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR'03, LNCS 2850*, pages 422–436, September 2003. Almaty, Kazakhstan.

[11] J.Y. Girard. Linear Logic: its Syntax and Semantics. In J.Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995.

[12] S. Ishtiaq and P. O'Hearn. BI as an assertion language for mutable data structures. In *28th ACM Symposium on Principles of Programming Languages, POPL 2001*, pages 14–26, London, UK, 2001.

[13] C. Kreitz and J. Otten. Connection-based theorem proving in classical and non-classical logics. *Journal of Universal Computer Science*, 5(3):88–112, 1999.

[14] P. O'Hearn, J. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. In *15th Int. Workshop on Computer Science Logic, CSL 2001, LNCS 2142*, pages 1–19, Paris, France, 2001.

[15] P.W. O'Hearn and D. Pym. The Logic of Bunched Implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999.

[16] D.J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002.

[17] J. Reynolds. Separation logic: A logic for shared mutable data structures. In *IEEE Symposium on Logic in Computer Science*, pages 55–74, Copenhagen, Danemark, July 2002.