H. Comon , H. Ganzinger, C. Kirchner,
H. Kirchner, J.-L. Lassez , G. Smolka (editors):

Theorem Proving and Logic Programming with Constraints

Dagstuhl-Seminar-Report; 24
21.10.-25.10.91 (9143)

ISSN 0940-1121

Report on the Dagstuhl-Seminar

"Theorem Proving and Logic Programming with Constraints"

Organizers:
Hubert Comon (Orsay)
Harald Ganzinger (Saarbrücken)
Claude Kirchner (Nancy)
Hélène Kirchner (Nancy)
Jean-Louis Lassez (Yorktown Heights)
Gert Smolka (Saarbrücken)

October 21–25, 1991

The last few years have seen a considerable amount of successful research in logic programming and theorem proving based on constraints. Initially devised as a concept for enhancing Prolog by a logic version of arithmetics, the ideas of constraint logic programming have spread out to affect the thinking about many other problems in programming languages and theorem proving.

As can be seen from the abstracts of the talks, the workshop succeeded in bringing together researchers from different areas, including artificial intelligence, logic and functional programming, algebraic specification, unification, rewriting and theorem proving.

Constraints are logical descriptions (e.g., first-order formulae) for which specialized and relatively efficient reasoning techniques are available. Typically, constraint techniques exploit some kind of constraint propagation to avoid or reduce the combinatorial search coming with general purpose techniques.

Some of the talks were concerned with constraint techniques for particular domains or applications. Other talks outlined and analyzed general frameworks providing for the disciplined combination of constraint techniques with more general reasoning mechanism such as resolution. Seen from this perspective, the integration of constraints is in fact part of the more general research activity concerned with the combination and modularisation of computational logics.

This Dagstuhl workshop will be followed by a series of related workshops, which will be organized in Val d'Ajol (1992), Barcelona (1993) and Munich (1994) by the recently approved Esprit working group "Construction of Computational Logics" (CCL, #6028).

Gert Smolka (Saarbrücken)

Abstracts of the Talks

A Feature-based Constraint System for Logic Programming with Entailment

Hassan Aït-Kaci (Rueil-Malmaison), Andreas Podelski (Paris),
Gert Smolka (Saarbrücken)

We present the constraint system *FT*, which we feel is an intriguing alternative to Herbrand both theoretically and practically. As does Herbrand, *FT* provides a universal data structure based on trees. However, the trees of *FT* (called feature trees) are more general than the trees of Herbrand (called constructor trees), and the constraints of *FT* are finer grained and of different expressivity. The basic notion of *FT* are functional attributes called features, which provide for record-like descriptions of data avoiding the overspecification intrinsic in Herbrand's constructor-based descriptions. The feature tree structure fixes an algebraic semantics for *FT*. We will also establish a logical semantics, which is given by three recursive axiom schemes fixing the first-order theory *FT*.

FT is a constraint system for logic programming, providing a test for unsatisfiability, and a test for entailment between constraints, which is needed for advanced control mechanisms.

The two major technical contributions of this paper are (1) an incremental entailment simplification system that is proved to be sound and complete, and (2) a proof showing that *FT* satisfies the so-called "independence of negative constraints."

Efficient AC1-Matching Using Constraints

J. Avenhaus, J. Denzinger, T. Hoffmann (Kaiserslautern)

In equational reasoning it has been proved powerful to apply inference steps modulo a fixed theory. Among the most prominent theories are AC, AC1 and ACI, where A, C and I refer to associativity, commutativity and idempotency, respectively, and 1 refers to the axiom $f(x, 1) = x$. We are interested in efficient matching algorithms modulo these theories since matching is needed for simplification and so is to be performed very often. Notice that AC1 and ACI are collapsing theories.

Our strategy is based on the well-known idea to construct unification or matching algorithms by combining the rules Decomposition, Simplification and Isolation of alien subterms. We propose not to solve each subproblem directly but to approximate the solutions in form of *intervals*. (The definition of *interval* depends on the underlying theory). These constraints (i) are easy to propagate and (ii) cut down the search space for a global solution drastically. This allows one to

- early detect that no solution exists
- early find one (or all) solution(s)
- represent the set of solutions in a compact form for further usage.

Constrained Resolution with Redundancy Criteria

Leo Bachmair (Stony Brook), Harald Ganzinger (Saarbrücken)

We extend previous results about resolution with ordering restrictions and selection functions to the case of general (quantifier-free) first-order formulas with constraints. The refutation completeness of our calculi is compatible with a general and powerful redundancy criterion which includes most (if not all) techniques for simplifying and deleting formulas. The spectrum of first-order theorem proving techniques covered by our results includes constrained resolution, theory resolution, ordered resolution, positive resolution, hyper-resolution, semantic resolution, and set-of-support resolution, as well as their extension to general first-order formulas. An additional feature in the latter case is our efficient handling of equivalences as equalities on the level of formulas.

Solving Perfect Squares Placement in CHIP

N. Beldiceanu (Orsay)

In this presentation we introduce a new constraint for finite domains: the cumulative constraint. With this constraint we show how to solve difficult placement and scheduling problems: perfect squares placement, rectangles placement, scheduling with cumulative and precedence constraint, and disjunctive scheduling. The perfect squares placement problem attempts to pack a set of squares of different size into a larger square, in such a way, that no hole or overlapping occurs. We present this problem and the corresponding CHIP program and heuristic used. For the three other problems we present the problem statement and the corresponding results obtained.

Logic Programming with Pseudo-Boolean Constraints

Alexander Bockmayr (Saarbrücken)

We introduce a new constraint logic programming language CLP(PB) for logic programming with pseudo-Boolean constraints. The language is an instance of the general constraint logic programming scheme CLP(X) proposed by Jaffar and Lassez in 1987. Pseudo-Boolean constraints are equations or inequalities between pseudo-Boolean functions. A pseudo-Boolean function is an integer-valued function $f : \{0,1\}^n \rightarrow \mathcal{Z}$ of Boolean variables. Pseudo-Boolean functions occur in many application areas, in particular in operations research. An interesting connection to logic is that inference problems in propositional logic can be translated into linear pseudo-Boolean optimization problems. More generally, pseudo-Boolean constraints can be seen as a way of combining two of the most important domains in constraint logic programming: arithmetic and Boolean algebra. We present variable elimination algorithms for pseudo-Boolean unification and unconstrained pseudo-Boolean optimization. Both algorithms subsume the well-known Boolean unification algorithm of Büttner and Simonis and can be implemented in a similar way.

A WAM Extension for Type-Constraint Logic Programming and its Correctness Proof

Egon Börger (Pisa)
[joint work with Christoph Beierle (Stuttgart)]

Based on Börger's and Rosenzweig's formal derivation of Warren's Abstract Machine for executing Prolog (see LNCS 533) from Börger's Prolog Algebras (see LNCS 440, 452), we provide a mathematical specification of a WAM extension to type-constraint logic programming and prove its correctness. We keep the notion of constraints rather abstract to show that our definitions apply to many constraint formalisms like Prolog III or CLP(R). We demonstrate the method on a concrete system, the Protos Abstract Machine (PAM), an extension of the WAM by polymorphic order-sorted unification as required by the logic programming language Protos-L.

We use Gurevich's notion of evolving algebra. First we refine Börger's Prolog algebras (developed for standard Prolog) to Protos-L algebras, essentially by replacing unification by general constraints systems and by consequently refining the Prolog algebra functions "unify" and "subres" to "solution" and "conres" (in the spirit of the Prolog III algebras of Börger & Schmitt, LNCS 533). Starting from these Protos-L algebras we then specify the PAM by a sequence of evolving algebras extensions, each representing a refinement level. We keep the type term constraints abstract as long as possible in order to be able to carry over the corresponding refinement steps from the WAM, together with their correctness proof. The final specification of type terms allows to introduce special PAM optimizations (like switching w.r.t. type constraints).

Summing up we have a proof for the main theorem: PAM is correct w.r.t. Protos-L (for *each* compiler satisfying some precisely specified natural conditions).

Temporal Logic Programming with Metric and Past Operators

Christoph Brzoska (Karlsruhe)

Temporal logic allows to use logic programming to specify and to program dynamically changing situations and non-terminating computations in a natural and problem oriented way. Recently so called metric or real-time temporal logics have been proposed for specification of real-time systems, where not only qualitiative but also quantitative temporal properties are very important. In this work we investigate a subset of metric temporal Horn logic called *MTL-programs*, for which we give a translation into CLP($\mathcal{A}'$)-programs and CLP($\mathcal{A}'$)-goals over a suitable algebra $\mathcal{A}'$. We give a restriction of the CLP($\mathcal{A}'$)-derivation mechanism sufficient for derivation of MTL-goals from MTL-programs, which admits efficient satisfiability checking of the constraints generated. Its worst case complexity is linear in the number of variables involved contrary to general satisfiability checking of constraints over $\mathcal{A}'$, which is NP-complete.

Constraints and Restricted Quantifiers

Hans-Jürgen Bürckert (Saarbrücken)

The basic idea of constrained resolution is replacing the unification procedure of Robinson's Resolution Principle by constraint solving procedures. We assume to have clauses whose variables are constrained by open formulae of a distinguished signature. These constraints act as filter for the assignments of the variables and are to be interpreted over a specified constraint theory, which is given as a consistent set of constraint axioms or somewhat more general as a class of constraint models. The variable assignments of the constraints into to these models may be seen as solutions of the constraints. Now, a constrained resolution step is like a classical resolution step, but instead of unification of argument terms one has to combine the constraints of the parent clauses and

to check solvability of the combined constraint. Constrained resolution provides then a sound and complete refutation calculus in the following sense: A set of constrained clauses is unsatisfiable w.r.t. the constraint theory iff for each constraint model there exists a derivation of an empty but still constrained clause, such that the constraint of the empty clause is solvable in that model.

There are also some better results, if the constraint theory satisfies certain requirements. For instance, if it is specified by a set of first order axioms, then a set of clauses is unsatisfiable iff finitely many empty constrained clauses can be derived, such that the existentially closed disjunction of their constraints is a logical consequence of the constraint theory. If the constraint theory is specified by a set of definite clauses and the constraints are conjunctions of atoms only, then a single empty clause provides a refutation of an unsatisfiable set of clauses as it is the case for classical resolution procedures.

We present a general framework for constraint simplification and show that constrained resolution together with constraint simplification is again a sound and complete refutation calculus. Notice, that in the above results we did not require to solve the constraints, but just to check for solvability. Constraint simplification, now, may use transformation into "solved forms" for constraints. Examples are obtained if we consider unification of terms as a constraint simplification process that transforms equations into simpler or solved equations, namely the unifiers.

Finally we will discuss how this constrained clause logic may be extended to a more general logic by taking constraints as quantifier restrictions for both universal quantifiers (constraints of a clause can be seen as restrictions of the implicite universal quantifiers of the clause) and existential quantifiers. The main problem here is the question of Skolemization of restricted quantifiers and here in turn the problems lie in the fact that quantifier restriction may have no solution in some constraint models (empty quantification). From the operational point of view Skolemization brings in new function symbols and this leads to question of how to extend constraint checking with these new Skolem functions. This is a non-trivial problem and cannot be solved in general, but just with respect to specific constraint theories or classes of constraint theories. We consider this question for the task of combining so-called cartesian constraint systems (where constraints always constrain single variables, e.g., like it is the case for sort constraints) and equational constraints, i.e., E-unification problems. We describe the combined constraint solving procedure for these problems under certain restrictions. Instances are sorted E-unification problems and the combination of E-unification and concept constraints, which are cartesian constraints over KL-ONE style concept description languages (known as terminological logics).

Constraint Solving and Model Construction

Ricardo Caferra, Nicolas Zabel (Grenoble)

A method is proposed to systematize the simultaneous search for a refutation and for Herbrand models of a given conjecture. It is based on an extension of resolution using equational problems and the inference system included in the method is proved to be sound and refutationally complete. For some classes of formulas the method is indeed a decision procedure. In particular it is a decision procedure for the Bernays-Schönfinkel class (a class for which no resolution term ordering strategy is known to be a decision procedure). Some examples of model construction—including one for which other resolution-based decision procedures fail to detect satisfiability—are developed in detail.

The method is also useful in cases in which model construction is not required. The search space in resolution-based deductions can be greatly decreased. This is shown in solving a question-answering problem, considered to be hard.

Models are built by constructing relations on Herbrand universe. The relationship between these models and finite ones is established. The class of these constructible relations is precisely characterized. Some of the rules introduced in order to extend resolution are essentially new. It is proved that they are necessary for enlarging the class of models the method is able to build. A brief comparison with existing methods which bear similarity with ours, *either* in the use of constraints *or* in the search of a model, shows the originality of our proposal. Experiments have been performed using our theorem prover and a system handling disequations implemented at the University of Kaiserslautern. Some hints about directions for extending the class of formulas for which models can be constructed are given.

Completion of Rewrite Systems with Membership Constraints

Hubert Comon (Orsay)

We consider a constrained equational logic where the constraints are membership conditions $t \in \underline{s}$ where $\underline{s}$ is interpreted as a regular tree language. Our logic includes a fragment of second order equational logic (without λ-expressions) where second order variables range over regular sets of contexts. The problem with constrained equational logics is the failure of the critical pair lemma. This is the reason why we propose new deduction rules for which the critical pair lemma is restored. Computing critical pairs requires, however, to solve some constraints in a second-order logic with membership constraints. This is the most difficult result of the paper: we give a terminating set of transformation rules for these formulas, which decides the existence of a solution, thus showing a new fragment of second order logic in which unification is decidable.

Since an order-sorted signature is nothing but a bottom-up tree automaton, order-sorted equational logic falls into the scope of our study; our results show how to perform order-sorted completion without regularity and without sort decreasingness. It also shows how to perform unification in the order-sorted case, with some higher-order variables (without any regularity assumption).

A New Perspective on Integrating Functional and Logic Languages

John Darlington, Yi-ke Guo, Helen Pull (London)

Traditionally the integration of functional and logic languages is performed by attempting to integrate their semantic logics in some way. Many languages have been developed by taking this approach, but none manages to exploit fully the programming features of both functional and logic languages and provide a smooth integration of the two paradigms. We analyse the main existing approaches for language integration to reveal the source of these inadequacies. We propose that improved integrated systems can be constructed by taking a broader view of the underlying semantics of logic programming. A novel integration language paradigm, constraint functional logic programming (CFLP), is then proposed. CFLP unifies functional and logic programming features systematically in terms of constraints. CFLP generalises constraint logic programming (CLP) by admitting user-defined functions via a purely functional subsystem of a CLP language, and enhances the expressive power of functional languages by exploiting a solving capability over functional programs. Moreover, CFLP offers the possibility of designing a promising declarative concurrent programming system by generalising concurrent logic programming, data flow programming and conventional shared memory-based concurrent programming models in various aspects.

An Approach to Constraint Functional Logic Programming

Francisco Javier López Fraguas, Mario Rodríguez-Artalejo (Madrid)

We present a general scheme CFLP(X) for first-order constraint functional logic programming which plays, with respect to functional logic languages with constructor discipline, a similar role to the well known of CLP(X) with respect to Horn clause

logic programming. In CFLP(X), over a base structure equipped with a set of predefined functions and predicates, we define new ones by means of constrained conditional rewriting rules. We formulate a declarative semantics in a general setting, where base structures are Scott domains and functions and predicates are continuous, and we obtain a complete characterization of the minimal model of a program as the least fixpoint of an associated operator. Finally, we propose a sound and complete operational semantics for the case in which base structures are flat cpo's and all functions and predicates are strict. (A technical report describing this work is available.)

Introducing Simplification Rules
(Towards a framework for extensible constraint logic programming)

Thom Frühwirth (München)

We are investigating the use of logic programs to define constraints. Constraints and their simplification should be definable by clauses. Being able to represent constraints in the same formalism as the rest of the program greatly facilitates the extension, prototyping and comparison of constraint systems. We propose to specify constraints by predicates, which are defined by definite Horn clauses as usual. To specify how constraints simplify we propose multi-headed flat guarded clauses called *Simplification Rules* (SiRs). As a result a tight integration of the logic programming and constraint solving component in a constraint logic programming language is achieved. Beyond the conceptual argument in favour of our approach, reasoning about Simplification Rules allows to prove correctness as well as termination and confluence of constraint simplification.

Refutation Theorem Proving for Hierarchical First-Order Theories

Harald Ganzinger (Saarbrücken)
[joint work with Leo Bachmair (Stony Brook), Uwe Waldmann (Saarbrücken)]

In this work we extend previous results by Bachmair and Ganzinger (1991) on theorem proving for first-order clauses with equality to hierarchical first-order theories. Semantically such theories are confined to conservative extensions of the base models. It is shown that superposition together with base variable abstraction and constraint refutation is refutationally complete for sufficiently complete theories. For the proof we introduce a general concept of approximation between refutation proof systems. This allows us to reduce the problem to the known case of (flat) first-order theories.

Logic Programming, Equations, and Deductive Planning

G. Grosse, S. Hölldobler, J. Schneeberger (Darmstadt)

Classical Logic was devised for the representation of static knowledge. First attempts to model actions and changable situations within Classical Logic were kind of clumsy and led to a massive criticism on a logical foundation of Artificial Intelligence. In the meantime there are a variety of logical approaches for modelling actions and situations. Modal and temporal logics, Bibel's linear connection method, and Girard's linear logic are among these approaches. The key idea of linear logic is the introduction of so-called *linear* connectives, for which the weakening and contraction rule are no longer applicable. Hence, literals connected by linear connectives represent ressources. They cannot be reproduced via weakening and contraction and are used up if required as a precondition of an action.

In this talk we present an equational logic programming language, which contains linear operators on the term level. With the help of several examples in the domain of deductive planning we illustrate the properties of this language. In particular, we show that the underlying calculus is equivalent to the linear connection method and linear logic.

Unification with Polymorphic Type Constraints

Michael Hanus (Dortmund)

We propose a typed logic where the type structure allows the combination of parametric and subtype polymorphism. Since the subtype order is specified by Horn clauses for the inclusion relation $\leq$, it is possible to define type constructors which are monotonic as well as anti-monotonic in their arguments. For instance, parametric order-sorted type structures for logic programs with higher-order predicates can be specified in our framework. Proof procedures like resolution for this typed logic require a unification procedure on well-typed terms. We describe such a unification procedure by a set of transformation rules which generate a set of type constraints from a given unification problem. We show how these type constraints can be solved for particular type structures.

Global Constraint Satisfaction

Walter Hower (Koblenz)

The constraint satisfaction problem (CSP) deals with the assignment of values to variables according to existing constraints. Given n variables with their finite domains, the admissible combinations of values form a set of n-tuples which represents the globally consistent solution.

The procedure (for the synthesis of the n-ary (constraint) relation) which is commonly cited is already more than twelve years old—an eternity in our rapidly growing AI area.

A different approach to try to tackle the (exponential) complexity (which is inherent in the problem regarding both space and time) has been published in the "Artificial Intelligence" journal (in 1989). However, the technique depicted there only slightly improves the space complexity and is even worse in its time behaviour.

By contrast with both methods mentioned above, the algorithm I present here has a drastically better time complexity and yields a considerable improvement regarding the demand on memory. Therefore, it may be considered as a further development of the current CSP techniques known so far. Additionally, our procedure allows a direct access to a partially parallel processing.

Higher Order Constraint Logic Programming

Timothy J. Hickey (Waltham)

In this paper we present a family of higher order constraint logic programming languages, CLP*(D), parameterized by constraint domains D. In these languages, predicates are first class objects in the sense that they are viewed as objects of a constraint domain which can appear in constraints and which can also be applied to terms. In line with this philosophy, all predicates are denoted by variables and predicate definitions are viewed as constraints which bind the predicate variable to the appropriate term. Since a predicate definition is a constraint, one can nest predicate definitions and thereby obtain a lexically scoped, block structured language which facilitates modular programming styles. In CLP*, we also merge logical, functional, and set-theoretic programming styles by relying on the fact that the set of predicates on D^n is in one-to-one correspondence with the set of subsets of D^n and also with the set of multi-valued, partial functions from D^{n-1} to D. Thus, from a semantics viewpoint the only difference between predicates, sets, and multi-valued partial functions is in the notation used to define and apply them. In CLP* we allow all three notations to be used interchangeably. This feature of the language allows one to program at a conceptually high level, and the resulting code compiles to efficient CLP programs. Our principal example of the versatility of this language is a CLP* program which implements a Prolog interpreter directly from the lattice-theoretic definition of the meaning of a Prolog program as the least fixed point of the T-operator on the powerset of the Herbrand Universe. This CLP* program compiles into the classical "solve interpreter." Our compilation strategy is to first define a naive compiler from CLP* to CLP and then to apply a CLP partial evaluator to optimize the resulting code. This method shows promise of being a practical use of partial evaluation. The CLP* languages can be thought of as providing a user-interface to CLP languages which allows higher order predicate, functional, and set syntax to be used without incurring the overhead of higher order unification.

Order-sorted Rewriting and Completion in G-algebra

Claude Kirchner, Hélène Kirchner (Villers-lès-Nancy)

The notion of G-algebra, recently developed by A. Megrelis, for handling semi-functions, equalities, subsets, and inclusion is a logic which has several characteristics that differentiate it from previous order-sorted approaches: each item of information is coded as a formula and typing is proving (not only parsing). This logic is complete with respect to the considered class of models and each class of models has a free algebra.

We propose an operational semantics for the deduction in G-algebra, in which sort computation and equality deduction are performed at the same time, by introducing the notion of decorations. Decorations are sets of sorts, recording the currently proved sorts of a term. This gives ability to prove equational theorems of the form $t = t'$ but also typing theorems of the form $t : s$. These proofs are performed using decorated rewrite rules that perform equational replacement and decoration rewrite rules that enrich the decorations and record sort information. Completeness of rewriting with respect to deduction is obtained via a completion process where superposition and rewriting are performed on decorated terms. Thus in this framework, the restrictions of regularity, coherence and sort-decreasingness are not needed any more to get the usually expected (and needed) results.

Polynomial and Elementary Constraints

Pierre Lescanne (Vandœuvre-lès-Nancy)

After a survey on how polynomial and elementary[1] constraints can be used to prove termination of term rewrite systems, it was explained that the problem of proving termination of rewrite systems by number theoretic interpretations boils down to prove that a function $F(X_1, \ldots, X_n)$ is positive over the naturals i.e., over N^n. This problem is known to be undecidable even if one restricts to polynomials and it is interesting to consider polynomial over R_+^n if one wants to get decision procedures. In the talk three methods were presented.

- *A naive method for proving that a polynomial is positive* was first expound. This method is obviously not a decision procedure, but works well in all the practical situations.
- *A decision procedure for positiveness of multivariate polynomials over* R_+^n was then presented. It is based on real algebraic geometry, especially Sturm-Habicht method and sub-resultants. It uses geometric properties of algebraic curves and representations of algebraic numbers.
- *A method for proving positiveness of elementary functions* which is an extension of the naive method for polynomials was described.

[1] *Elementary* should be taken in the sense of number theoretic functions, i.e., function that are built in our case by composition of additions, multiplications and exponentiations.

Completion of First-order Clauses with Equality by Basic Superposition with Ordering Constraints

Robert Nieuwenhuis (Barcelona)
[joint work with Albert Rubio]

We discuss the formalism of constrained clauses and its use in order to importantly reduce the search space in completion of first-order clauses with equality, making it also possible to obtain complete sets in more cases.

First we explain how we apply *equality constraints* for proving the completeness of *basic superposition*: a restricted form of superposition in which only the subterms *not* generated in previous inferences are superposed upon. This allows to compute far less inferences in Knuth-Bendix completion procedures for equations and other first-order clauses with equality and also in other paramodulation-based theorem provers. Our result can be seen as the counterpart in superposition of the proof of completeness of *basic narrowing* in narrowing techniques.

Second, we extend the techniques to further restrict inference systems by the use of *ordering constraints*. These constraints keep information about the maximality of literals and terms in previous inferences. Future inferences with choices of maximal literals/terms that are incompatible with the constraints can then be shown to be unnecessary. Our results extend and improve previous work by Peterson for the equational case, since no additional inference rules are needed and we can deal with full first-order clauses. Moreover, our methods for dealing with ordering constraints are compatible with basic superposition and with the notions of *redundancy* of clauses and inferences as defined by Bachmair and Ganzinger. The satisfiablity problem for this kind of constraints has recently been shown to be decidable by Comon.

Constraint Systems Corresponding to Nonclassical Logics

Hans Jürgen Ohlbach (Saarbrücken)

Modal Logics are used in various interpretations, as epistemic logics, action logics, temporal logics etc. Many of these interpretations require the modal operators to be parametrized with agents, actions and the like. A generalization of the possible worlds semantics of modal logic in a way that new interpretations become possible is presented. Furthermore correlations between different parameters can be incorporated into the logic in the same way as particular properties of the accessibility relation such as reflexivity, transitivity etc. in the classical case are incorporated into the logic. One interpretation which becomes possible is that of a probability logic. A formula $[.5]F$ may for example express "in at least 50% of all cases (worlds) F holds".

A first order version of the logic is defined and a set of elementary operations on the parameters is derived from basic operations on the possible worlds semantics.

For this logic there is a 'compiler' for translating formulae into predicate logic. This compiler translates the characteristic axiom schemas which describe the correlations between parameters into first-order axioms which then give rise to particular constraint theories.

Constraint Programming based on Relative Simplification

Gert Smolka (Saarbrücken)

Constraint logic programming, concurrent logic programming and negation as failure are three well-established subareas of logic programming that developed more or less independently. More recently it turned out that at least constraint and concurrent logic programming can be profitably merged into one more general paradigm, now becoming known under the name concurrent constraint programming. We argue that negation also fits well into this new paradigm, and that in fact the operational semantics for deep guards in concurrent logic programming has much in common with constructive negation, a powerful operational semantics for negation in logic programming.

We present a rewrite calculus that gives a unified and abstract operational account of concurrent constraint languages with disjunction and negation. The calculus is parameterized with respect to a constraint theory and a program extending the constraint system with new predicates. Computation amounts to rewriting expressions according to the rules of the calculus. The major inovation of the calculus is the principle of relative simplification, which simplifies a constraint at a position P in an expression E modulo its context, which is a constraint uniquely determined by P and E. The principle of relative simplification at the same time provides for constraint simplification, incremental entailment checking, deep guards, and constructive negation.

Hierarchical Equational Problems

Ralf Treinen (Saarbrücken)

A *Hierarchical Equational Problem* (hep) is presented as $P = (\Sigma_P, \Sigma_E, E)$, where $\Sigma_P \subseteq \Sigma_E$ are signatures and E is a set of Σ_E-equations. Let $[[P]]$ be the free functor defined by P, that is for any algebra $\mathcal{A} \in Alg_{\Sigma_P}$

$$[[P]]\mathcal{A} := \mathcal{T}(\Sigma_E \setminus \Sigma_P, \mathcal{A})/E$$

is the quotient by E of the free Σ_E-algebra generated by $\mathcal{A}$. Given a first-order Σ_E-sentence w, a first-order Σ_P-sentence v is a *P-weakest parameter condition* (P-wpc) of w, if

$$\text{for all} \quad \mathcal{A} \in Alg_{\Sigma_P}: \quad \mathcal{A} \models v \quad \Leftrightarrow \quad [[P]]\mathcal{A} \models w$$

$P = (\Sigma_P, \Sigma_E, E)$ is *wpc-complete* if a P-wpc exists for each Σ_E-sentence, and *effectively wpc-complete* if we can compute a P-wpc for any Σ_E-sentence.

The wpc-completeness of a hep is a question of expressiveness of first order logic. We show that the wpc-completeness of a particular hep is strongly related to the compactness of an associated logic.

The effective wpc-completeness of a hep with $\Sigma_P = \emptyset$ corresponds to the decidability of non-hierarchical equational problems. We show that the effective wpc-completeness of a hep P is equivalent to the decidability of the theory of $[[P]]\mathcal{A}$ relative to the theory of $\mathcal{A}$ for all $\mathcal{A} \in Alg_{\Sigma_P}$.

Besides the case $E = \emptyset$ we give a class of effectively wpc-complete heps with non-trivial E. It is essential that for this class of heps there always exist Σ_P sentences that for each algebra $A \in Alg_{\Sigma_P}$ describe the co-domains of the functions in $[[P]]\mathcal{A}$.

Solving Equality and Subtree Constraints

Sauro Tulipani (Camerino)

We consider the problem of solving, in algebras of rational or infinite trees, systems of equalities, inequalities or $t \leq s$, $t \not\leq s$, where $\leq$ is interpreted as the *subtree relation.* We call *singular* a signature with no more than one symbol of positive arity. We denote by $RT[X]$, $IT[X]$ the algebras of rational and infinite trees, respectively, where X is any set of new (free) constant symbols; we simply write RT, IT if X is empty. Moreover, if $\mathcal{A}$ is a structure, we denote by $Th_E(\mathcal{A})$ the set of first order existential sentences which are true in $\mathcal{A}$. Then we prove the following results.

1. *$Th_E(RT[x_1], \leq) = Th_E(RT[X], \leq) = Th_E(IT[X], \leq)$ for every non-empty $X = \{x_1, \ldots\}$. We give an algorithm to decide this set of sentences.*

2. *Moreover, $Th_E(RT[x_1], \leq) = Th_E(RT, \leq) = Th_E(IT, \leq)$ if the signature is not singular.*

3. *$Th_E(RT[x_1], \leq) \neq Th_E(RT, \leq) = Th_E(IT, \leq)$ if the signature is singular. We give also an algorithm to decide the last set of sentences.*

These results are related to previous results of A. Colmerauer, M.J. Maher, H. Comon and P. Lescanne, G. Marongiu and S. Tulipani, K.N. Venkataraman.

Venkataraman proved also that, for some signature, the fragment $\exists\forall_{\leq}$ of sentences with prefix of existential quantifiers followed by bounded (to $\leq$) universal quantifiers, is undecidable. Treinen put Venkataraman's argument to a more general setting by proving analogous results for infinite tree algebras.

We denote by Δ_0 the set of formulas where all the quantifiers are bounded by $\leq$ and we denote by $\exists\Delta_0$ the set of sentences which are existential quantification of Δ_0-formulas. Then, (with G. Marongiu) we have obtained by the method of interpretations of the natural numbers into tree algebras.

4. *Venkataraman's and Treinen's undecidability results.*

5. $Th_{\exists\Delta_0}(RT, \leq) \neq Th_{\exists\Delta_0}(IT, \leq)$. *In fact,* $Th_{\exists\Delta_0}(RT, \leq)$ *is* r.e. *and* $Th_{\exists\Delta_0}(IT, \leq)$ *has degree of unsolvability not less than* Σ_1^1.

Dagstuhl-Seminar 9143: List of Participants (update: 30.03.92)

Jürgen **Avenhaus**
Universitat Kaiserslautern
FB Informatik
Postfach 30 49
W-6750 Kaiserslautern
Germany
avenhaus@informatik.uni-kl.de

Egon **Börger**
Istituto di Scienze Dell'Informazione
Università di Pisa
Corso Italia 40
56100 Pisa
Italy
boerger@dipisa.di.unipi.it

Hans-Jürgen **Bürckert**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
hjb@dfki.uni-sb.de

Leo **Bachmair**
SUNY at Stony Brook
Applied Mathematics
Stony Brook NY 11794
USA
Leo@sbcs.sunysb.edu

Nicolas **Beldiceanu**
COSYTEC S.A.
4 rue Jean Rostard
F-91893 Orsay Cedex
France

Alexander **Bockmayr**
Max-Planck-Institut für Informatik
Im Stadtwald 15
W-6600 Saarbrücken 11
Germany
bockmayr@mpi-sb.mpg.de

Christoph **Brzoska**
SFB 314
Universitat Karlsruhe
Postfach 69 80
W-7500 Karlsruhe 1
Germany
brzoska@ira.uka.de

Ricardo Caferra
LIFIA-IMAG
46 avenue Felix Viallet
F-38031 Grenoble Cedex
France
caferra@neptune.imag.fr or: caferra@lifia.imag.fr
tel.: +33-7657-4659 (4805)

Alain **Colmerauer**
Groupe d'Intelligence Artificielle
Faculté des sciences de Luminy
70 route Léon Lachamp
F-13288 Marseille Cedex 9
France

Hubert **Comon**
Université de Paris Sud
LRI Bat. 490
F-91405 Orsay Cedex
France
comon@sun8.lri.fr
tel.: +33-1-69 41 66 35

Rachid **Echahed**
LIFIA-IMAG
46 avenue Felix Viallet
F-38031 Grenoble Cedex
France
echahed@lifia.imag.fr
tel.: +33-76 57 46 67 (76 51 46 00 ext. 5044)

Thom **Frühwirth**
ECRC München
Arabellastr. 17
W-8000 München 81
Germany
thom@ecrc.de
tel.: +49-89-92699-140

Harald **Ganzinger**
Max-Planck-Institut für Informatik
Im Stadtwald 15
W-6600 Saarbrücken 11
Germany
hg@mpi-sb.mpg.de
tel.: +49-681-302-5361 (Sekretariat: 5360)

Steffen **Hölldobler**
TH Darmstadt
FB Informatik/FB Intellektik
Alexanderstr. 10
W-6100 Darmstadt
Germany
steffen@intellektik.informatik.th-darm-stadt.de
tel.: +49-6151-16 5469

Marianne **Haberstrau**
Université de Paris Sud
LRI Bat. 490
F-91405 Orsay Cedex
France
haberstr@sun8.lri.fr

Michael **Hanus**
Max-Planck-Institut für Informatik
Im Stadtwald 15
W-6600 Saarbrücken 11
Germany
michael@mpi-sb.mpg.de

Timothy J. **Hickey**
Dept. of Computer Science
Ford Hall
Brandeis University
Waltham MA 02254-9110
USA
tim@cs.brandeis.edu
tel.: +1-617-736-2706

Walter **Hower**
Universitat Koblenz - Landau
Institut für Informatik
Rheinau 3 - 4
W-5400 Koblenz
Germany
walter@uni-koblenz.de
tel.: +49-261-9119 426 (secr. 432)

Jieh **Hsiang**
SUNY at Stony Brook
Applied Mathematics
Stony Brook NY 11794
USA
hsiang@sbcs.sunysb.edu

Jean-Pierre **Jouannaud**
Université de Paris Sud
LRI Bat. 490
F-91405 Orsay Cedex
France
jouannaud@margaux.inria.fr
tel.: +33 1 69 41 69 05

Claude **Kirchner**
INRIA Lorraine and CRIN
Campus Scientifique
BP 239
F-54506 Vandoeuvre lès Nancy
France
ckirchne@loria.loria.fr

Hélène **Kirchner**
CRIN and INRIA Lorraine
Campus Scientifique
BP 239
F-54506 Vandoeuvre lès Nancy
France
hkirchne@loria.crin.fr or: hkirchne@loria.loria.fr

Pierre **Lescanne**
CRIN (CNRS) & INRIA-Lorraine
BP 239
F-54506 Vandoeuvre-lès-Nancy Cedex
France
lescanne@loria.fr

Robert **Nieuwenhuis**
Universidad Politécnica de Cataluña
Dept. L.S.I.
Pau Gargallo 5
E-08028 Barcelona
Spain
roberto@lsi.upc.es

Hans Jürgen **Ohlbach**
Max-Planck-Institut für Informatik
Im Stadtwald 15
W-6600 Saarbrücken 11
Germany
ohlbach@mpi-sb.mpg.de

Fernando **Orejas**
Universidad Politécnica de Catalunqa
Dept. L.S.I.
Pau Gargallo 5
E-08028 Barcelona
Spain
orejas@lsi.upc.es
tel.: +34-3-401 70 18

Andreas **Podelski**
Digital Equipment Corporation
Paris Research Laboratory
85 Avenue Victor Hugo
F-92500 Rueil-Malmaison Cedex
France
podelski@prl.dec.com

Laurence **Puel**
Université de Paris Sud
LRI Bat. 490
F-91405 Orsay Cedex
France
puel@margaux.inria.fr

Mario **Rodrìguez-Artalejo**
Universidad Complutense de Madrid
Departamento de Informática y
Automática
E-28040 Madrid
Spain
MARIO@emducm11.bitnet

Michaël **Rusinowitch**
INRIA-Lorraine and CRIN
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy Cedex
France
rusi@loria.crin.fr

Gert **Smolka**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
smolka@dfki.uni-sb.de

Jörg **Sueggel**
Universitat Bielefeld
Technische Fakultat
Postfach 86 40
W-4800 Bielefeld
Germany
joerg@techfak.uni-bielefeld.de
tel.: +49-521-106 2925

Ralf **Treinen**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
treinen@dfki.uni-sb.de
tel.: +49-681-302-5314

Sauro **Tulipani**
Dipartimento di Matematica e Fisica
Universitàggg di Camerino
62032 Camerino
Italy
tulipani@camvax.cineca.it

Uwe **Waldmann**
Max-Planck-Institut für Informatik
Im Stadtwald 15
W-6600 Saarbrücken 11
Germany
uwe@mpi-sb.mpg.de
tel.: +49-681-302 5431