mj-Prolog
— 1. Proof Theoretical Foundations —

Rodrigo Readi Nasser*
Universität München

09.11.1999

In this article we give the proof theoretical foundations of an extension of Prolog, to be called **mj-Prolog**, for dealing with representations of recursive functions, as Prolog deals with terms. In opposition to other related systems, mj-Prolog remains in the framework of first order intuitionistic logic, recursive functions are determined and represented by equations that on one side have the usual interpretation in first order logic and on the other side can be used as term reduction system for evaluating the function. Section 1 deals with proof theoretical foundations of classical logic programming, but emphasis is given only in the logical part that is common to mj-Prolog, not in the search of unknown terms, because in it lies the main difference with mj-Prolog. Section 2 deals with equations systems for specifying unknown recursive functions, the solving of these systems in mj-Prolog corresponds to the unification in Prolog. Section 3 brings the matter of sections 1 and 2 together, it shows how logic programming can be used for stating systems of equations that specify recursive functions.

1 Foundations of Classical Logic Programming

In this section we deal with proof theoretical foundations of *classical* logic programming, classical in the sense that we do not deal with search of other objects than terms, only a short note at the end of §9 points to the use of unknowns for delaying the selection of terms to be found later by unification, lifting in this classical context is not treated, it may be found in [16]. From the logical point of view, we are dealing with the hereditary Harrop formulae introduced to logic programming as a generalization of horn clauses in [12], but we deal with it in the context of Gentzen's calculus NJ and Prawitz' normal form, not in the context of LJ and the the uniform proofs of [12]. We deal with intuitionistic predicate logic, but subsection 1.5 deals with a restriction of hereditary Harrop formulae that suffices for expressing classical predicate logic with the help of negation axioms, and subsection 1.6 deals with a further restriction in which intuitionistic and classical logic coincide, the generalized horn clauses treated there and

*Lehrstuhl für Mathematische Logik, Mathematisches Institut, Zimmer 417, Theresien-Str. 39, 80333-München. readi@rz.mathematik.uni-muenchen.de

originally from [17] are a generalization of the horn clauses in the original Prolog. The equality axioms in section 2 are generalized horn clauses.

1.1 Gentzen's Natural Deduction

§1

Terms, formulae, sequents

We consider **open formulae** recursively built starting from *open atomic formulae* with the following **logical symbols**: $\supset$ for implication, $\wedge$ for conjunction, $\vee$ for disjunction, $\forall$ for universal quantification, $\exists$ for existential quantification. An **open atomic formula** is built with an n-ary **predicate symbol** of the language, perhaps 0-ary, and n *open terms*. We have two infinite sets of auxiliary 0-ary function symbols in the language: symbols for **variables** to be bound by quantifiers and symbols for **free variables** denoting *arbitrary constants*. **Open terms** are recursively built with variables, free variables and other **function symbols** of the language. A **formula** is an open formula that does not contain any variable not bound by a quantifier. A **term** is an open term not containing any variable. Free variables are allowed to occur in formulae and terms. A **sequent** $\Sigma \vdash \alpha$ is a pair consisting of a (finite, perhaps empty) list Σ of formulae, its **antecedent** containing its **assumptions**, and a formula α, its succedent. All symbols we are considering are from a set L, a **language** for the predicate logic.

§2

The calculus NJ for negationless logic

The following calculus corresponds to Johanssons NM in [5] if we allow a 0-ary predicate symbol $\Box$ denoting contradiction. Although Gentzen's intuitionistic calculus NJ for natural deduction has three schemata more for the negation, we continue using the name NJ.

$$\text{GS} : \frac{}{\Sigma \vdash \alpha};$$

$$\text{FE} : \frac{\Sigma \cup \{\alpha\} \vdash \beta}{\Sigma \vdash \alpha \supset \beta}; \qquad \text{FB} : \frac{\Sigma \vdash \alpha, \Sigma \vdash \alpha \supset \beta}{\Sigma \vdash \beta};$$

$$\text{UE} : \frac{\Sigma \vdash \alpha, \Sigma \vdash \beta}{\Sigma \vdash \alpha \wedge \beta}; \qquad \text{UB}_1 : \frac{\Sigma \vdash \alpha \wedge \beta}{\Sigma \vdash \alpha}, \text{UB}_2 : \frac{\Sigma \vdash \alpha \wedge \beta}{\Sigma \vdash \beta};$$

$$\text{OE}_1 : \frac{\Sigma \vdash \alpha}{\Sigma \vdash \alpha \vee \beta}, \text{OE}_2 : \frac{\Sigma \vdash \beta}{\Sigma \vdash \alpha \vee \beta}; \qquad \text{OB} : \frac{\Sigma \vdash \alpha \vee \beta, \Sigma \cup \{\alpha\} \vdash \gamma, \Sigma \cup \{\beta\} \vdash \gamma}{\Sigma \vdash \gamma};$$

$$\text{AE} : \frac{\Sigma \vdash \varphi[q]}{\Sigma \vdash \forall v \varphi[v]}; \qquad \text{AB} : \frac{\Sigma \vdash \forall v \varphi[v]}{\Sigma \vdash \varphi[t]};$$

$$\text{EE} : \frac{\Sigma \vdash \varphi[t]}{\Sigma \vdash \exists v \varphi[v]}; \qquad \text{EB} : \frac{\Sigma \vdash \exists v \varphi[v], \Sigma \cup \{\varphi[q]\} \vdash \gamma}{\Sigma \vdash \gamma}.$$

Conditions:

- The succedent α of the undersequent $\Sigma \vdash \alpha$ of GS should be a member of the antecedent Σ.
- The expressions $\Sigma \cup \{\alpha\}$, $\Sigma \cup \{\beta\}$ and $\Sigma \cup \{\varphi[q]\}$ in the OB-, FE- and EB-schemata represent the new antecedents obtained by appending the formulae α, β, and respectively $\varphi[q]$ to Σ, these formulae are the assumptions **discharged** by the OB-, FE- or EB-rule.
- $\varphi[v]$ in AE, AB, EE and EB represents an open formula, $\varphi[s]$ represent a formula obtained by substituting the variable v in $\varphi[v]$ by the term s, where s is q or t.
- The q in AE and EB represents a free variable called **proper variable** of the rule, this free variable q should appear neither in $\varphi[v]$ nor in a formula of the undersequent of the rule.
- The expression t in AB and EE represents a term.
- The proper variable q of an AE- or EB-rule, or the term t of an AB- or EE-rule is the only **auxiliary term** of the rule, the other rules have no auxiliary term.

E (Einführung) stays for the **introduction rules**, and B (Beseitigung) for the **elimination rules**. F (folgt) stays for $\supset$, U (und) for $\wedge$, O (oder) for $\vee$, A (all) for $\forall$, E (existiert) for $\exists$. In Prawitz' natural deduction [15], the English names $\supset I$, $\supset E$, $\&I$, $\&E$, $\vee I$, $\vee E$, $\forall I$, $\forall E$, $\exists I$, $\exists E$ are used instead of Gentzen's German names FE, FB, UE, UB, OE, OB, AE, AB, EE, EB.

The antecedent of an oversequent of a rule is either identical to the antecedent of the undersequent or contains a formula more, the formula discharged by a OB-, FE- or EB-rule. Gentzen [4] and Prawitz [15] consider natural deduction as derivations with NJ, but writing down only the succedents, keeping track of the original assumptions (in GS-rules) and of their discharging in the deduction process. In later work, Gentzen considered NJ as a sequent calculus like his LJ.

§3

Unambiguous proper variables

An NJ-composition Ξ has **unambiguous proper variables** if proper variables of different rules in it are formally different and each proper variable appears at most in sequents above the undersequent of its rule (of course excluding this undersequent). Renaming the proper variables of such a NJ-composition with new (not appearing in Ξ) symbols for free variables, formally different proper variables with formally different symbols, leads again to a NJ-composition with unambiguous proper variables.

If, for example, q appears as proper variable of two rules and in the endsequent, then a single renaming would replace q by a q' that would also appear in these three places. In spite of it, following 3.10, page 198 in [4], one can find for every NJ-composition Ξ a similar NJ-composition Ξ' with unambiguous proper variables and with the same endsequent, so that each firstsequent of Ξ' is obtained by renaming the symbols for free variables in a corresponding firstsequent of Ξ (all occurrences of a q in the sequent with the same q'). The NJ-composition Ξ' is found recursively on the number of nodes in Ξ; for the oversequents of the rule r in Ξ whose undersequent is the endsequent of Ξ one has shorter NJ-compositions (subtrees of Ξ), and hence one can find NJ-compositions of them with unambiguous proper variables; since these proper variables can be renamed, one can demand that proper variables appearing in different compositions also be different and do not appear in the undersequent of r (endsequent of Ξ), these NJ-compositions together with r form the desired NJ-composition Ξ'. The Ξ' obtained in this way is, independent of the symbols for free variables selected in the process, unique up to renaming of proper variables.

Since we consider rules with different auxiliary terms as different, even if they are formally equal, we also consider the occurrence of a proper variable in an auxiliary term as an occurrence in the rule, even if the auxiliary term is not part of the oversequent or undersequent of the rule. For example, this may happen with an EE- or AB-rule whose oversequent coincide with its undersequent. Because of this, the renamings on the above recursive procedure must be also done in the auxiliary terms of such pathological rules.

§4

NJ-Validity structural rules

We say that a rule is **NJ-valid**, if from NJ-derivations of its oversequents one can build an NJ-derivation of its undersequent. A schema is **NJ-valid**, if all its rules are NJ-valid.

We consider now schemata of the form

$$\frac{\Sigma \vdash \xi}{\Sigma' \vdash \xi},$$

where (1) Σ' is obtained by inserting some formulae in Σ, (2) Σ' is obtained by exchanging the order of the formulae in Σ and (3) Σ' is obtained by deleting a formula of Σ that is formally equal to another. In the first case the schema is called, according to [4], **Verdünnung**, in the second **Vertauschung** and in the third **Zusammenziehung**. These are the three schemata for **structure rules (in the antecedent)**

Every structure rule is NJ-valid. A derivation of its oversequent $\Sigma \vdash \xi$ is easily transformed into a derivation of its undersequent $\Sigma' \vdash \xi$ by making some changes in each sequent, from the endsequent to the leaves, so that each rule in the derivation is transformed into a similar rule of the same schema. Only in the case of Verdünnung, it is necessary to first transform the NJ-derivation of $\Sigma \vdash \xi$ into one with unambiguous proper variables and then to rename the proper variables, so that no one occurs in a formula added to Σ for obtaining Σ'.

1.2 Definite and Goal Formulae, Decomposition of Definite Formulae

§5

D- and G-Formulae, P-sequents

Using the concepts introduced in [15], pages 15 and 43, we define (open) **D-formulae** as the (open) formulae not containing a (open) *positive subformula* whose *principal sign* is $\exists$ or $\vee$, and (open) **G-formulae** as the (open) formulae not containing a (open) *negative subformulae* whose *principal sign* is $\exists$ or $\vee$. An (open) **A-formula** is an atomic one. An equivalent definition for defining open D- and G-formula can be given as in [6] by simultaneous recursion:

$$D := A|\forall v D|D_1 \wedge D_2|G \supset D,$$
$$G := A|\exists v G|\forall v G|G_1 \vee G_2|G_1 \wedge G_2|D \supset G.$$

An open formula is simultaneously an open D-formula and an open G-formula if and only if it contains neither $\exists$ nor $\vee$. A **P-sequent** is a sequent whose succedent is a G-formula and whose assumptions are D-formulae. In these definitions introduced in [6], D stays for “definite”, G for “goal” and P for “program”.

§6

D-, G- and T-sequences

A **partial D-sequence** for a D-formula φ begins with φ and is recursively built using the following rules for finding a successor for a formula:

- An atomic formula has no successor.
- A successor of $\forall v \psi[v]$ is a formula of the form $\psi[t]$, where t is a selected **auxiliary term**.
- A successor of $\psi_1 \wedge \psi_2$ is ψ_1 or ψ_2; we say that the first or respectively that the second subformula was selected.
- The successor of $\psi_1 \supset \psi_2$ is ψ_2, the formula ψ_1 is the **eliminated** formula.

Each step leads from a D-formula to a new D-formula, we will never reach a formula having $\vee$ or $\exists$ as principal sign, we can hence continue until reaching an atomic formula. A partial D-sequence ending with an atomic formula is a **D-sequence**. Each eliminated formula is a G-formula, the sequence of them is the **G-sequence** of the (partial) D-sequence. The sequence of selected auxiliary terms is its **T-sequence**.

Two D-sequences are **equal** if and only if they begin with the same formula and were constructed following the same steps. Otherwise they are different, even if they are *formally equal*, this may happen when there is a formula in the D-sequence of the form $\forall v \psi[v]$ where $\psi[v]$ is not explicitly dependent on v or a formula of the form $\psi \wedge \psi$.

§7

Unknowns for delaying the selection of the T-sequence

Two D-sequences are **similar** if they begin with the same formula and, with the only possible exception of the selected auxiliary terms, were constructed following the same steps. There are finitely many similarity classes of D-sequences for a φ.

If we admit in our signature infinitely many new auxiliary 0-ary functional symbols to be called **unknowns**, we can construct D-sequences for φ by only selecting new unknowns as auxiliary terms. Such D-sequences are **most general**. Every D-sequence is obtained by substituting the unknowns used as auxiliary terms in a similar most general D-sequence, and its G-sequence is obtained with the same substitution. Every element of a similarity class can be obtained by such substitutions in a most general representant.

1.3 The Calculus mj for P-sequents

§8

The m-schema

Let Ξ be a tree composed with a GS-rule and some FB-, AB- and UB-rules, so that its firstsequents are the left oversequents of its FB-rules and the rest of its sequents builds a thread beginning with the undersequent $\Sigma \vdash \xi$ of the GS-rule and ending with a sequent having an atomic formula as succedent, the endsequent of the tree. Let Δ be the list of succedents of the thread. Then, Δ is a D-sequence for ξ, the list of succedents of the firstsequents of the tree is the G-sequence of Δ, all the sequents of the tree have Σ as antecedent. In this way we have a bijective correspondence between such trees and the pairs (Σ, Δ) consisting of a list Σ of formulae and a D-sequence Δ for a D-formula ξ in Σ. We can summarize the whole tree with a rule $m(\Sigma, \Delta)$ all of whose sequents have Σ as antecedent, whose undersequent has the last formula of Δ as succedent, the list of succedents of its oversequents is the G-sequence of Δ. The terms in the T-sequence of Δ are the **auxiliary terms** of the rule $m(\Sigma, \Delta)$. For different Σ or different Δ we consider the corresponding $m(\Sigma, \Delta)$ to be different m-rules, even in the case of formal equality.

§9

The calculus mj

If Σ consist of D-formulae, then an $m(\Sigma, \Delta)$ rule consists of P-sequents. If the undersequent of an m-, FE-, UE-, OE-, AE- or EE-rule is a P-Sequent, then also are its oversequents. The rules of the **calculus mj** are the m-, FE-, UE-, OE-, AE- and EE-rules consisting of P-sequents. Only m-, AE- and EE-rules of mj have auxiliary terms. It is a calculus for building deductions of P-sequents, to be applied backwards to reduce the derivability of a P-sequent to the derivability of other P-sequents, its correctness with respect to NJ is provided in the paragraph above.

The schema for a reduction is determined by the principal sign of the succedent of the P-sequent: the m-schema is for atomic succedents, FE, UE, OE, AE and EE for succedents with principal sign $\supset$, $\wedge$, $\vee$, $\forall$ or respectively $\exists$. If the principal sign is $\supset$ or $\wedge$, there is only one possible reduction with an FE- or UE-rule. If it is $\vee$, then there are two possible reductions with OE-rules. If it is $\forall$, then there are infinitely many reductions with AE-rules, but they are equivalent up to renaming the proper variable. If it is $\exists$, then there is one for each selected auxiliary term, but the selection can be postponed using an unknown. If it is atomic, then there is one for each selected D-sequence, but postponing the selection of the auxiliary terms in the T-sequence by taking unknowns, there are first finitely many possibilities. This is the **analytic property** of mj.

§10

The subformula property of mj, tracing symbols, proper variables of a sequent

The formulae α and β are **immediate subformula** of the larger formulae $\alpha \wedge \beta$, $\alpha \vee \beta$ and $\alpha \supset \beta$. The formula $\varphi(t)$, where t is a term, is an **immediate subformula** of the larger formulae $\forall v \varphi(v)$ and $\exists v \varphi(v)$. The formula ψ is a **subformula** of φ, if there is a sequence of formulae beginning with φ and ending with ψ, such that the successor is an immediate subformula of the predecessor. — Every symbol appearing in an oversequent of an mj-rule appears either in its undersequent or in an auxiliary term of it. Every formula of an oversequent of an mj-rule is a subformula of its undersequent. This is the **subformula property** of mj, it is shared with LJ, but not with NJ.

We have the concept of the proper variable of an mj-rule. The **proper variables** of an mj-composition tree are the proper variables of its mj-rules. For defining the concept of proper variables of a sequent in such a tree we need the concept of a thread. A **thread** in an mj-composition is a sequence $A_0, A_1, \dots A_n$ of sequents in it, such that every A_{i+1} is an oversequent of the rule of the mj-composition having A_i as undersequent; these rules are also considered part of the thread. There is exactly one thread beginning with the endsequent of an mj-composition and ending with a given sequent S in it, the subformula property implies that every symbol of the given sequent S can be traced until an auxiliary term of a rule in this thread or until the endsequent; the list Q of proper variables of AE-rules in this thread, in the order they appear in the thread, is called the **list of proper variables** of S; the symbols for free variables in this thread that can be traced until the endsequent or until an auxiliary term of an EE- or m-rule are called the **proper arbitrary constants** of S.

The definition and results in §3 holds also for mj. If an mj-composition has unambiguous proper variables, then the list of proper variables of a sequent in it consists of formally different symbols.

1.4 The Calculus mj and Prawitz' Normal Form

§11

Long normal form and mj-derivations

The segments of a normal deduction in the sense of [15] containing neither OB- ($\vee$ E) nor EB-rules ($\exists$ E) have only one formula. A derivation not containing these rules and whose minimum segments in its paths are atomic is said to be in **long normal form**.

Our mj can be seen as a calculus for representing deductions in long normal form of P-sequents. A deduction in mj can be transformed into a deduction in long normal form by substituting the m-rules with the corresponding combinations of GS (assumption), FB ($\supset$ E), UB ($\wedge$ E) and AB ($\forall$ E). The E-parts of its paths correspond to the threads by which the $m(\Sigma, \Delta)$-rules have been substituted, the minimum segments in its paths correspond to the atomic succedents of the undersequents of the m-rules. The converse transformation is also possible, a deduction in long normal form of a P-sequent can be recursively transformed into an mj-derivation: the E-part of each path π containing the endsequent is a D-sequence to be substituted by an m-rule, the smaller deductions above the minor premises of FB-rules ($\supset E$) whose major premises are on π are substituted by corresponding deductions in mj (obtained by the recursive process).

§12

Deductions of P-sequents in Prawitz' normal form

A normal deduction of a G-formula from a set of D-formulae does not contain $\exists E$ or $\vee E$ rules. Every assumption in such a deduction, discharged or not, is a D-formula. Consequences of E-rules in the deduction are D-formulae, consequences of I-rules are G-formulae. Minimum formulae of its paths are D- and G-formulae at the same time (i. e. they contain neither $\exists$ nor $\vee$). We prove this by induction on the size of such a natural deduction Π of a G-formula from a set Γ of D-formulae. If the end-formula E of Π is the conclusion of an E-rule, then a path π containing E has no minor premise of an $\exists E$ or $\vee E$ rule; otherwise, as in the proof of corollary 6 of [15], the major premise M of this rule would be a strictly positive subformula of a formula in Γ (namely, of the first formula of a path containing it, this first formula cannot be discharged in Π because M is in the E-part of the path and below M there is no $\supset I$ rule), but this cannot be the case because the principal sign of M is $\exists$ or $\vee$. Hence, π is the only path in Π containing the end-formula E, its first formula is in Γ (as above, it cannot be discharged), every formula in π is a D-formula because it is a strictly positive subformula of the first formula, π contains neither minor nor major premises (the latter are not D-formulae) of $\exists E$ or $\vee E$. We can now decompose Π in the path π and the subtrees of Π above minor premises of $\supset E$ rules whose major premises are in π; these minor premises are G-formulae (because the major premises are D-formulae) and the assumptions of these subtrees are subsets of Γ; by inductive hypothesis, the proposition holds for these subtrees, and hence also for Π. All that remains is to prove the proposition for the case that the end-formula E of Π is the conclusion of an I-rule. By deleting the end formula, we get a smaller deduction, or two in the case where this formula is the conclusion of $\wedge I$, whose end-sequent is a G-formula and whose assumptions are D-formulae (the assumptions are the same of Π, except in the case that the end-sequent is $D \supset G$ where the D-formula D is added). Since for the smaller deductions the proposition holds by inductive hypothesis, it also holds for Π.

§13

Completeness of mj

For proving the completeness of mj with respect to NJ it is enough to find a deduction in long normal form for each NJ-derivable P-sequent. Since for the NJ-derivable P-sequent there is always a normal deduction, it is enough to prove that this deduction can be transformed into a deduction in long normal form, and for this we use the results of the paragraph above.

A normal deduction of a G-formula from a set of D-formulae can be transformed into a normal deduction whose paths have atomic minimum formulae. For proving this, we enlarge the paths of the original deduction as done in Lemma 6.5.3, page 154 of [18]. The minimum formula M of a path does not contain $\vee$ or $\exists$. If the minimum formula is of the form $F_1 \supset F_2$, we can add F_1 to the assumptions, enlarge the E-part by adding F_2 at its end and enlarge the I-part by deducing $F_1 \supset F_2$ from F_2 discharging F_1. If the minimum formula is of the form $\forall vF$, we can enlarge the E-part by deducing $F[a/v]$ with $\forall E$, where a is a new symbol for free variables, and enlarge the I-part by deducing $\forall vF$ with $\forall I$. If the minimum formula is of the form $F_1 \wedge F_2$, we can duplicate the subtree above this formula, in one of the copies enlarge the E-part by deducing F_1 with $\wedge E$, in the other copy enlarge it by deducing F_2, and then we can paste both trees by deducing $F_1 \wedge F_2$ with $\wedge I$. One can begin enlarging each path π containing the end-formula, and continue with the paths of the subtrees above the minor premises of $\supset I$ rules whose major premises are in π, the termination of this process is proved by double induction, on the depth of the deduction and on the total amount of connectives in the minimum formulae of the paths π.

§14

mj-Derivations and LJ-derivations, Uniform Proofs

The introduction rules FE, UE, OE, AE and EE of mj are identical to the introduction in succedent rules FES, UES, OES, AES and EES of Gentzen's LJ. The rule $m(\Sigma, \Delta)$ of mj, where Δ begins with $\xi \in \Sigma$ and ends with A, can be seen as a composition of a GS-rule, some rules for introduction in the antecedent FEA, UEA and AEA, and a "Zusammenziehung" rule forming a thread, so that the active formulae of the introduction rules in the antecedent forms a sequence corresponding to the reverse of Δ, so that the "Zusammenziehung" rule contracts the last formula ξ of this sequence with a copy of it present from the beginning. Of course, some details should be fixed by selecting an appropriate variant of LJ. This enables us to build a correspondence between mj-derivations and a class of LJ-derivations similar to the correspondence between mj-derivations and deductions in long normal form introduced in §11. This class of LJ-derivations corresponds to the **uniform proofs** in [6]. It should be no surprise, since in the context of uniform proofs calculi similar to mj were stated, in [7] one for a fragment of intuitionistic linear logic. Behind this correspondence is the well known correspondence between normal NJ and normal LJ proofs, see for example [15] or [13] for more details.

1.5 The $\{\supset, \forall\}$-Segment

§15

Head and body

For a list V of variables and an open formula ξ we denote with $V\xi$ the open formula obtained by putting in front of ξ a block of universal quantifiers with the variables of V, in the same order they appear in V. For (possibly empty) lists $V_n, \ldots, V_1, V_0$ of variables and a list $\xi_n, \ldots, \xi_1, \xi_0$ of open formulae, we define the list $\eta_0, \ldots, \eta_n$ of open formulae recursively, so that η_0 is $V_0\xi_0$ and η_{k+1} is $V_{k+1}(\xi_{k+1} \supset \eta_k)$. With

$$V_n\xi_n \& \cdots \& V_1\xi_1 \supset V_0\xi_0$$

we denote the last defined open formula η_n. Of course, for $n = 0$ this open formula is $V_0\xi_0$.

An (open) $\{\supset, \forall\}$-formula is one not containing other logical signs than $\supset$ and $\forall$, it is an (open) D- and G-formula at the same time. It is not difficult to see that every open $\{\supset, \forall\}$-formula ξ can be expressed in a unique way as $V_n\xi_n \& \cdots \& V_1\xi_1 \supset V_0\xi_0$ with an atomic ξ_0. In this case, the list $\xi_n, \ldots, \xi_1$ is denoted by body(ξ) and called **body** of ξ, the open atomic formula ξ_0 is denoted by head(ξ) and called **head** of ξ. The universal quantifiers binding the variables in $V_n, \ldots, V_1, V_0$ are the **principal quantifiers** of ξ; they can be moved, perhaps after a renaming, to the left for obtaining an equivalent open formula whose principal quantifiers build a block at front of it.

§16

Instantiation

A principal quantifier of a $\{\supset, \forall\}$-formula can be deleted and the variables bound by it substituted by a term for obtaining a new formula implied by the original. For a $\{\supset, \forall\}$-formula ξ and a list T of terms, one defines the **instance** $T * \xi$ of ξ as the $\{\supset, \forall\}$-formula obtained by deleting principal quantifiers in ξ, from left to right, and substituting the variables bound by them by the terms of T, from left to right, until all principal quantifiers are deleted or until there are no more terms in T for the rest of the principal quantifiers. The instance $T * \xi$ can also be recursively defined by the following equations:

$$\begin{aligned}
[\,] * \xi &= \xi, \\
T * A &= A, \\
(R \cdot S) * \xi &= S * (R * \xi), \\
T * (\eta \supset \zeta) &= \eta \supset (T * \zeta), \\
[t] * (\forall v \zeta) &= \zeta^{v \to t},
\end{aligned}$$

where $[\,]$ denotes the empty list, A an atomic formula, $R \cdot S$ the concatenation of R and S, $[t]$ the list containing only t, and $\zeta^{v \to t}$ the formula obtained by substituting v by t.

The variables substituted by a term t in this instantiation process correspond to the elimination of a universal quantifier with the auxiliary term t in the process of building a D-sequence. The process of building a D-sequence Δ for a $\{\supset, \forall\}$-formula ξ depends only on the selected T-sequence T that has as much elements as principal quantifiers in

ξ, the last formula in Δ is head$(T*\xi)$, the corresponding G-sequence body$(T*\xi)$. And conversely, if T has so much terms as principal quantifiers in ξ, then one can construct a D-sequence Δ with T as T-sequence, so that head$(T*\xi)$ is the last element in Δ and body$(T*\xi)$ the corresponding G-sequence.

§17

m-, d- and g-rules

An $\{\supset,\forall\}$**-sequent** is one consisting of only $\{\supset,\forall\}$-rules. If the undersequent of an mj-rule is a $\{\supset,\forall\}$-sequent, then also its oversequents. Hence, restricting mj-schemata to $\{\supset,\forall\}$-sequents does not alter mj-derivability of $\{\supset,\forall\}$-sequents. FE-, AE- and m-rules restricted to $\{\supset,\forall\}$-sequents are denoted by $d(\Sigma\vdash\varphi)$, $g(\Sigma\vdash\varphi,q)$ and $m(\Sigma,\xi,T)$, where $\Sigma\vdash\varphi$ is the undersequent of the FE- and AE-rules, q the proper variable of the AE-rule, ξ and T the first formula end the T-sequence of Δ in the $m(\Sigma,\Delta)$-rule.

§18

Negation Axioms, classical predicate logic, an intuitionistic segment

Let $\Box$ be a 0-ary predicated symbol representing **contradiction**. As usual, we can paraphrase the negation $\neg\xi$ of a formula ξ with $\xi\supset\Box$, and any usual connective of classical logic using $\neg$, $\supset$ and $\forall$. For each predicate symbol R let wj_R be a formula of the form $\forall\bar{v}(\Box\supset R(\bar{v}))$ and wk_R a formula of the form $\forall\bar{v}(((R(\bar{v})\supset\Box)\supset\Box)\supset R(\bar{v}))$. Let wj be a list (the set) built with all the wj_R and wk with wk_R, these are the sets of intuitionistic and classical **negation axioms**. Let Σ be the list of *axioms of a theory*, of *additional postulates* in the sense of the calculus in [8], page 82 (or a similar one). The formulae φ derivable in this calculus coincide with the ones such that $wk\cup\Sigma\vdash\varphi$ is mj-derivable, or equivalently, NJ-derivable. The proof consists, as expected, in checking that the rules of one calculus is valid in the other. A special rôle plays the NJ-derivability of $wk\cup\Sigma\vdash\varphi$, where φ is a formula $((\xi\supset\Box)\supset\Box)\supset\xi$ of the schema (postulate) 8° of [8]. We see this by reducing the sequent with mj: after d- and g-reductions it is sufficient to add $(\xi\supset\Box)\supset\Box$ and every formula in body$(Q*\xi)$ to the antecedent and consider the succedent head$(Q*\xi)$; the latter formula is atomic with a predicate symbol R and a list of arguments T, after an m-reduction considering $T*\mathrm{wk}_{R,\Box}=((\mathrm{head}(Q*\xi)\supset\Box)\supset\Box)\supset\mathrm{head}(Q*\xi)$ it is sufficient to consider the succedent $(\mathrm{head}(Q*\xi)\supset\Box)\supset\Box$; after a d-reduction it is sufficient to add head$(Q*\xi)\supset\Box$ to the antecedent and consider the succedent $\Box$; after an m-reduction considering the first formula added to the antecedent it is sufficient to consider the succedent $\xi\supset\Box$; after a d-reduction it is sufficient to add ξ to the antecedent and consider the succedent $\Box$; after an m-reduction considering the formula head$(Q*\xi)\supset\Box$ in the antecedent it is sufficient to consider the succedent head$(Q*\xi)$; after an m-reduction considering the formula ξ in the antecedent it is sufficient to consider succedents from body$(Q*\xi)$; all these formulae are in the set of assumptions, an m-reduction for each of them confirms their derivability.

Although similarly the mj-derivability of wj $\vdash\Box\supset\xi$ holds, the above argument does not hold for intuitionistic logic: not all usual logical symbols can be paraphrased with $\neg$, $\supset$ and $\forall$ in intuitionistic logic. Using Prawitz' Normal form (or Gentzen's

Hauptsatz), we can prove that a sequent $\Sigma \vdash \varphi$ containing no other logical symbols than $\neg$, $\supset$ and $\forall$ is intuitionistically derivable if and only if, after paraphrasing $\neg$ with $\Box$ and $\supset$, $\text{wj} \cup \Sigma \vdash \varphi$ is mj-derivable.

§19

Restart as an alternative to contradiction axioms

Among all possible mj-reductions of a sequent $\Sigma \vdash A$ with an atomic A and Σ containing the axioms wk, there is always the one with the m-schema and the appropriate wk_R leading to $\Sigma \vdash (A \supset \Box) \supset \Box$; this latter sequent can only be reduced to $\Sigma \cup \{A \supset \Box\} \vdash \Box$ with the d-schema; this sequent should be reduced with the m-schema, for this a $\xi \in \Sigma \cup \{A \supset \Box\}$ and an appropriate T, so that $\text{head}(T * \xi)$ and $\Box$ coincide, are necessary; the formula $A \supset \Box$ is one of such formulae ξ, the set Σ may contain other such formulae, especially the ones of the form $B \supset \Box$ added to Σ in reductions with the m-schema using a wk_R; now it is clear what the possible reductions are, this leads to the following remark: adding the contradiction axioms wk is equivalent to the introduction of the **restart-rule**, this rule allows to reduce sequents $\Sigma \vdash A$ whose succedent A is atomic to sequents $\Sigma \vdash B$, where B is either $\Box$ or the atomic succedent of an "ancestor" sequent $\Pi \vdash B$ of the proof process, of course, reductions of $\Sigma \vdash A$ with the m-schema remain possible. — Adding the axioms wj is equivalent to the introduction of the similar rule that allows to replace an atomic succedent by $\Box$ in the sequent to be reduced.

The restart rule is implicit in traditional tableaux methods, it may be seen as a consequence of the fact that succedents of sequents in LK-rules may contain more than one formula. The use of a restart rule in intuitionistic tableaux methods is correct until some extent: all LK-schemata, with the exception of FES and NES, are valid in intuitionistic logic (FES and NES also if succedents contain no more than one formula). The restart-rule appears also in other proof procedures used today in logic programming and automatic theorem proving.

1.6 Generalized Horn Sequents

§20

Generalized horn clauses and sequents

A **generalized horn clause** is, as defined in [17], a formula having all its principal quantifiers as a block at the beginning, such that $\text{head}(\xi)$ is atomic and that all formulae in $\text{body}(\xi)$ are universal quantified atomic open formulae. A **generalized horn sequent** is a sequent $\Sigma \vdash \varphi$ consisting only of generalized horn clauses, whose antecedent Σ contains no free variable and whose succedent φ is a universal quantified atomic open formula. A generalized horn sequent $\Sigma \vdash \varphi$ can only be reduced with the m- and g-schemata to generalized horn sequents having the same antecedent Σ. From this and the validity of the restart rule follows: if $\Sigma \vdash \varphi$ of this form is derivable in classical logic, then either itself or $\Sigma \vdash \Box$ is derivable in minimal logic. Furthermore, if no formula in Σ has $\Box$ as head, then, as in the case of common SLD-Resolution and as noted in [17], minimal, intuitionistic and classical derivability of $\Sigma \vdash \varphi$ coincide.

§21

Contradiction and contradictions

Let Σ be a list of generalized horn formulae containing no free variable. Let Γ a list of universal quantified atomic open formulae, to be called as $\Box$ **contradictions**, and $\Gamma_\Box$ the list of generalized horn sequents obtained by substituting each γ in Γ by $\gamma \supset \Box$. If $\Sigma \vdash \gamma$ for a γ in $\Gamma \cup \{\Box\}$ is mj-derivable, then obviously also $\Sigma \cup \Gamma_\Box \vdash \Box$. Conversely, if $\Sigma \cup \Gamma_\Box \vdash \Box$ is mj-derivable, then also $\Sigma \vdash \gamma$ for a γ in $\Gamma \cup \{\Box\}$: the mj-derivation of $\Sigma \cup \Gamma_\Box \vdash \Box$ contains only sequents whose antecedent is $\Sigma \cup \Gamma_\Box$, either contains a rule $m(\Sigma \cup \Gamma_\Box, \gamma_\Box, T)$ whose $\gamma_\Box$ is in $\Gamma_\Box$ or not, in the latter case we can delete $\Gamma_\Box$ from the antecedents of the sequents in the mj-derivation for obtaining an mj-derivation of $\Sigma \vdash \Box$, in the former case we can find such an m-rule $m(\Sigma \cup \Gamma_\Box, \gamma_\Box, T)$, so that no other is in the subtree over its oversequent $\Sigma \cup \Gamma_\Box \vdash \gamma$, this subtree can be converted into an mj-derivation of $\Sigma \vdash \gamma$ by deleting $\Gamma_\Box$ in the antecedents of its sequents.

§22

Open m-rules and schemata

For a generalized horn clause ξ of the form

$$\forall \bar{v}(\forall \bar{v}_n A_n(\bar{v}, \bar{v}_n) \& \cdots \& \forall \bar{v}_1 A_1(\bar{v}, \bar{v}_1) \supset A_0(\bar{v}))$$

not containing free variables, where the A_i are open atomic formulae, for a list of open terms $\bar{t}(\bar{w})$ not containing free variables and whose variables are in $\bar{w}$, and for renamings $\bar{v}'_n, \ldots, \bar{v}'_1$ of the variables in the lists $\bar{v}_n, \ldots, \bar{v}_1$ not containing variables from $\bar{w}$, we define the rule

$$\frac{A_n(\bar{t}(\bar{w}), \bar{v}'_n), \ldots A_1(\bar{t}(\bar{w}), \bar{v}'_1)}{A_0(\bar{t}(\bar{w}))}$$

for deriving an atomic open formulae from atomic open formulae. We call the rules won in this way from ξ the **open m-rules** for (corresponding to) ξ, the schema for building the rules the **open m-schema**.

If ξ is in a list Σ, then substituting the overformulae $A_i(\bar{t}(\bar{w}), \bar{v}'_i)$ of the introduced open m-rule by the sequent $\Sigma \vdash \forall \bar{w} \forall \bar{v}'_i A_i(\bar{t}(\bar{w}), \bar{v}'_i)$ and the underformula $A_0(\bar{t}(\bar{w}))$ by the sequent $\Sigma \vdash \forall \bar{w} A_0(\bar{t}(\bar{w}))$ yields an mj-valid rule, a composition of an m-rule corresponding to ξ with g-rules under its undersequent and with g-rules followed by AB-rules (mj-valid) over its oversequents. This tells us, how to understand open m-rules: variables occurring in an open atomic formula of the rule should be seen as universally quantified in front of the formula, even if they are formally equal to variables in other open formulae of the rule; the formula ξ should be seen as a member of a list of axioms Σ, as in calculi like Hilbert's one.

A generalized horn sequent $\Sigma \vdash \forall \bar{w} A(\bar{w})$ not containing free variables is mj-derivable if and only if $A(\bar{w})$ is derivable with open m-rules corresponding to formulae in Σ. The above remark on the validity of the open m-rules confirms the contrapositive. In an mj-deduction of a generalized horn sequent we can group each m-rule with the g-rules that are immediately below its undersequent, we can transform the mj-derivation into a derivation with open m-rules recursively on the number of these groups, or of m-rules. Let

$$\frac{\Sigma \vdash \forall \bar{v}_n A_n(\bar{t}(\bar{q}), \bar{v}_n), \ldots \Sigma \vdash \forall \bar{v}_1 A_1(\bar{t}(\bar{q}), \bar{v}_1)}{\Sigma \vdash A_0(\bar{t}(\bar{q}))}$$

be the last m-rule in such a deduction, corresponding to $\xi \in \Sigma$, obtained by substituting the variables bound by the principal quantifiers by $\bar{t}(\bar{q})$. Below the undersequent $\Sigma \vdash A_0(\bar{t}(\bar{q}))$ are applications of g-rules generalizing at least the free variables in $\bar{q}$ that explicitly appear in $\Sigma \vdash A_0(\bar{t}(\bar{q}))$, we can add more g-rules in order that all free variables in $\bar{q}$ be substituted by variables $\bar{w}$, this can be done because at the end we will delete the quantifiers. In this m-rule we can delete the the antecedents $\Sigma \vdash$, delete the universal quantifiers $\forall v_i$, rename the $\bar{v}_i$ with $\bar{v}_i'$ not containing variables in $\bar{w}$, and substitute the $\bar{q}$ by $\bar{w}$ for obtaining an open m-rule with underformula $A_0(\bar{t}(\bar{w}))$ and overformulae $A_i(\bar{t}(\bar{w}), \bar{v}_i')$, here it is important that the formula ξ does not contain free variables, and hence no one in $\bar{q}$. We search now for derivations of these overformulae with open m-rules. We have smaller mj-derivations of each $\Sigma \vdash \forall \bar{v}_i A_i(\bar{t}(\bar{q}), \bar{v}_i)$, and after deleting the last g-rules, mj-derivations of $\Sigma \vdash A_i(\bar{t}(\bar{q}), \bar{q}_i)$ for a new list of free variables $\bar{q}_i$ not containing elements of $\bar{q}$, and after adding new g-rules, mj-derivations of $\Sigma \vdash \forall \bar{w} \forall \bar{v}_i A_i(\bar{t}(\bar{w}), \bar{v}_i')$. Since we only deleted and added g-rules, we have by inductive hypothesis the desired derivations.

§23

Herbrand models for generalized horn clauses

In this paragraph, as an exception, we admit model-theoretical argumentation. Let B_L be the set of atomic formulae (in the language L) containing no free variable (and as formulae, no variable). A herbrand **model** is a subset of B_L. After declaring some elements of B_L to be **contradictions**, we can say that a herbrand model is **consistent** if it contains no contradiction. The set A_L of terms containing no free variable (and no variable) is the **universum** of H. All herbrand models have the same universum, unless we expand the considered language L. For a list of terms $\bar{s}$ from A_L we also write $\bar{s} \in A_L$.

For a generalized horn clause ξ of the form

$$\forall \bar{v}(\forall \bar{v}_n A_n(\bar{v}, \bar{v}_n) \& \cdots \& \forall \bar{v}_1 A_1(\bar{v}, \bar{v}_1) \supset A_0(\bar{v}))$$

not containing free variables and a list $\bar{t}$ of terms in A_L for $\bar{v}$ we define the **a-rule** $a(\xi, \bar{t})$ by

$$a(\xi, \bar{t}) : \frac{A_k(\bar{t}, \bar{s}) : \bar{s} \in A_L, k = 1, \ldots, n}{A_0(\bar{t})}.$$

This rule $a(\xi, \bar{t})$ has the underformula $A_0(\bar{t})$. The atomic formulae of the form $A_k(\bar{t}, \bar{s})$, where $\bar{s}$ is a list of elements in A_L and A_k one of the open atomic formulae in body(ξ), are the overformulae of $a(\xi, \bar{t})$. The rule may be **infinitary**, it may have infinite many oversequents; but this is not the case when the $\bar{v}_k$ are empty lists, namely, when ξ is a *horn formula*. Open m-rules generate in a trivial way a-rules, all a-rules for a ξ can be obtained from the open m-rule whose open underformula is $A_0(\bar{w})$.

For a herbrand model H and a set Σ of generalized horn clauses not containing free variables, we say that H **models** Σ or that H is a **model of** Σ, if for each rule $a(\xi, \bar{t})$, where ξ is in Σ and the terms of $\bar{t}$ in A_L, the herbrand model H contains the underformula of $a(\xi, \bar{t})$ if it contains the overformulae. In particular, B_L models every such Σ. The intersection of arbitrary many models of a Σ is also a model of Σ. The **smallest** herbrand model of Σ is the intersection H_Σ of all models of Σ, it is a model of Σ, consistent if Σ has such one.

For a generalized horn sequent $\Sigma \vdash \varphi$ not containing free variables, we say that $\Sigma \models \varphi$ holds, if every model of Σ is also a model of $\{\varphi\}$. If $\Sigma \vdash \varphi$ is mj-derivable, then $\Sigma \models \varphi$ holds. For proving this, let us call an instance of an atomic open formula α, in which all variables were substituted by elements of A_L, an A_L-instance of α. It is easy to see that, if all A_L-instances of each open overformula of an open m-rule for a formula in Σ are in a herbrand model H of Σ, then also all A_L-instances of the open underformula. Hence, a model H of Σ contains all A_L-instances of an open formula α derived with the open m-rules for formulae in Σ. If $\Sigma \vdash \varphi$ is mj-derivable, φ is the closure of an open atomic formula α derivable with the open m-rules for formulae in Σ, and the sentence above says that a model H of Σ is also a model of $\{\varphi\}$: by definition, H models $\{\varphi\}$ if and only if it contains all A_L-instances of α.

From the paragraph above, we can conclude that H_Σ contains all elements in B_L that are mj-derivable from Σ. Contrary to the case for horn clauses, H_Σ may contain elements that are not derivable from Σ. Let L be a language, so that $A_L = \{a, b\}$ and $B_L = \{p(a), p(b), \Box\}$. Let $\Sigma = \{p(a), p(b), (\forall v p(v)) \supset \Box\}$. Then there are exactly three a-rules, one for each element in Σ; and since A_L is finite, they are even finitary. Now, $\Sigma \models \Box$ holds, $H_\Sigma = B_L$, but $\Sigma \vdash \Box$ is nor mj- nor NK-derivable. After adding a new 0-ary c to L and hence to A_L, a *witness* of the (non) validity of $\forall v p(v)$, the non-derivable element $\Box$ disappears from H_Σ.

2 Equations Systems Specifying Functions

In this section we use equations to specify unknown recursive functions, as linear systems of equations specify unknown numbers. Systems of equations do not necessarily completely specify its unknowns, neither in the case treated here nor in the case of linear equations: for example, some (dependent) unknowns could be put as functions of other (independent) ones to be later further specified. Linear equations are solved with Gaußtriangulization algorithm, the term-unification algorithm [11] can also be seen as a triangulization algorithm, we solve our systems of equations with a modification of the term-unification algorithm, the cp-completion introduced in subsection 2.4. The cp-completion algorithm can also be seen as a modification of Knuth-Bendix' algorithm, or as an algorithm for checking consistence of equations systems and finding term reduction systems as models for them: subsection 2.3 brings unification, completion and consistence checking together. Subsection 2.2 puts Term Reduction Systems [9] in the context of predicate logic. The relevance of dealing with these equations systems and of their solution lies in their immediate application to integrate logic and functional programming: we can substitute term unification in logic programming by cp-completion, as [6] does with higher order unification, but remaining in the framework of first order logic. Subsection 2.5 puts Term Unification [11] in the context of our equations systems.

2.1 Systems of Equations

§24

Terms and open terms

We consider in this section expressions denoting *functionals*. They are recursively built with 0-ary symbols, also denoting functionals, and a 2-ary symbol denoting the *application operator*. We distinguish three kinds of 0-ary symbols: (1) **ground symbols** (normally small letters like a, b, c); (2) **unknowns** denoting functionals to be *determined* by formulae and equations (normally capital letters like X, Y, Z, U); (3) **variables** supposed to *range* over expressions denoting functionals (normally small letters like u, v, w).

Our 0-ary symbols are *open terms*; if t_0 and t_1 are *open terms*, then (t_0t_1) is the *open term* denoting the application of t_0 on t_1. This is how every **open term** is built, our 2-ary application symbol is given by the parenthesis. Open terms not containing variables are also called **terms**.

As in [2], we may write (t_0) for an open term t_0, $(t_0t_1t_2)$ for $((t_0t_1)t_2)$, and recursively, $(t_0 \cdots t_{k-1}t_k)$ for $((t_0 \cdots t_{k-1})t_k)$. Eventually we may write $t_0 \cdots t_{k-1}t_k$, without blanks and outer parenthesis, or $t_0(t_1, \cdots, t_k)$ for $(t_0 \cdots t_{k-1}t_k)$. Every open term t can be expressed in a unique way in the form $(ft_1 \cdots t_n)$, where f consists of only a 0-ary symbol called **operator** of t; the open terms $t_1, \ldots, t_n$ are then called **arguments** of t, the number n is the **number of arguments** of t.

§25

Equations, instances of equations

Equations are open atomic formulae built with the 2-ary infix predicate symbol $=$, they have the form $s = t$, where s and t are open terms. Variables appearing in an equation are to be seen as universal quantified in front of the equation. We call s the **left part** of the equation $s = t$, t its **right part**.

A **closure** of an equation $s = t$ is a formula of the form $\forall \bar{w} s = t$, where $\forall \bar{w}$ is a block of universal quantifiers binding all variables in $s = t$. An **instance** of an equation is the equation obtained by substituting all occurrences of some variables by open terms, the same open term for each occurrence of the same variable in the equation.

A **ground** equation is one not containing variables; it is one of the form $s = t$, where s and t are terms.

§26

Equality axioms and systems of equations

We define the formulae **ref**, **sym**, **trans**, **cons** by

ref : $\forall v v = v$,
sym : $\forall u \forall v u = v \supset v = u$,
trans : $\forall u \forall v \forall w u = v \supset (v = w \supset u = w)$,
cons : $\forall u_1 \forall u_2 \forall v_1 \forall v_2 u_1 = u_2 \supset (v_1 = v_2 \supset (u_1 v_1) = (u_2 v_2))$;

the **decomposition** schema $\mathrm{dec}_{n,i}(f)$ by

$$\mathrm{dec}_{n,i}(f) : \forall u_1 \cdots \forall u_n \forall v_1 \cdots \forall v_n (f u_1 \cdots u_n) = (f v_1 \cdots v_n) \supset u_i = v_i,$$

where f is a term, n a positive natural number and i between 1 and n $(1 \leq i \leq n)$; and the similar **separation** schema

$$\mathrm{sep}_{n,i}((w r_1 \cdots r_n) = (w s_1 \cdots s_n)) :$$
$$\forall v_1 \cdots \forall v_m ((\forall w (w r_1 \cdots r_n) = (w s_1 \cdots s_n)) \supset (\forall w r_i = s_i)),$$

where $(w r_1 \cdots r_n) = (w s_1 \cdots s_n)$ represents an equation whose variables are in v_1, $\ldots$, v_n, w, whose two parts have the same number of arguments and the variable w as operator; thus, the r_i and s_i represent open terms eventually containing these variables.

The **equality axioms** are the formulae ref, sym, trans, cons, $\mathrm{dec}_{n,i}(f)$ for each ground f and appropriate pair (n, i), and sep. A **system of equations** consists of closures of some equations, called **equations of the system**, and the equality axioms. We are interested on the mj-derivability of closures of equations from systems. All formulae in a system are generalized horn formulae, more accurate: the last $\forall w$ in sep-formulae should be moved to the front, all the rest are horn formulae. Hence, a closure of an equation is mj-derivable from a system if and only if the equation is derivable from instances of equations of the system with the following *open m-schemata* (see §22) corresponding to the equality axioms:

$$\text{ref} : \frac{}{s = s}, \quad \text{sym} : \frac{r = s}{s = r}, \qquad \text{trans} : \frac{r = s,\ s = t}{r = t},$$

$$\text{cons} : \frac{r_1 = s_1,\ r_2 = s_2}{(r_1 r_2) = (s_1 s_2)},$$

$$\text{dec} : \frac{(f r_1 \cdots r_n) = (f s_1 \cdots s_n)}{r_i = s_i}, \quad \text{sep} : \frac{(w r_1 \cdots r_n) = (w s_1 \cdots s_n)}{(r_i = s_i)^{w \leftarrow t}}.$$

In dec, f is a ground symbol. In sep, w represents a variable, $(r_i = s_i)^{w \leftarrow t}$ is an instance of $r_i = s_i$ obtained by substituting the variable w by an open term t. We allow variables to occur in the equations of these schemata: this is the reason why we defined equations as open atomic formulae and not as their closure. The advantage of considering derivation trees with these schemata is that it is easier to see how to exchange the order of derivation rules.

Of course, we can take the same variable represented by w as t in sep, this would lead to a schema similar to dec, and the original sep would be a composition of this schema with the **instantiation schema** that produce the rules having an instance of its overequation as underequation: since we want all instantiations at the leaves of the deduction tree, we do not take the instantiation schema and keep our sep. Later, we will introduce a rule that, as Robinsons resolution [1] and as SLD-Resolution [10], delays instantiation, that instantiates with a most general unifier only when an instantiation is necessary.

We *use* systems of equations to *specify* its unknowns, to *define* the *manifold* of solutions for the unknowns, as $\{x^2 + y^2 + z^2 = 1, x + y + z = 0\}$ is used to define a circle. We say that the system S_1 **constrains** the system S_2, that S_2 is **more general** than S_1 and that $S_1 \vdash S_2$ holds if each formula in S_2 is derivable from S_1. We say that S_1 and S_2 are **equivalent** and that $S_1 \equiv S_2$ holds if $S_1 \vdash S_2$ and $S_2 \vdash S_1$ hold.

2.2 Term Reduction Systems

§27

Derivability of $r \to_n s$

Compositions of n times the trans-schema are equivalent to the schema

$$\text{trans}_{n+1} : \frac{s_0 = s_1,\ s_1 = s_2,\ \ldots,\ s_n = s_{n+1}}{s_0 = s_{n+1}}.$$

We define trans_0 as ref. If we compose the trans-schema with the cons-schema so that the underequation of trans is an overequation of cons, then we can move down the trans-schema for obtaining an equivalent composition (having endequation and first equations formally equal to the ones of the original tree) of twice the cons-schema, the ref-schema and the trans-schema, so that each overequation of trans is an underequation of cons, and an overequation of one cons is the underequation of ref. A composition using only ref-, trans- and cons-rules can be transformed into an equivalent one having the same number n of trans-rules, but all together near the root, and then the trans-rules can be substituted by a trans_{n+1}-rule whose underequation is the endequation.

A **derivation** of $s \to t$ from a system S, where s and t are open terms, is by definition a derivation of a closure of $s = t$ from the formulae of S excluding sym-, dec- and sep-formulae, or equivalently, a derivation of $s = t$ from instances of equations of the system with the ref, trans and cons open m-schemata. Such a derivation containing $n - 1$ trans-rules is a derivation of $s \to_n t$, and one containing only a ref rule (and hence with equal s and t) is a derivation of $s \to_0 t$. Depending on the existence of derivations we say that $s \to t$ or $s \to_n t$ is **derivable** from S, and that $S \vdash s \to t$ or $S \vdash s \to_n t$ holds.

It is obvious that, if $s' = t'$ is an instance of $s = t$ and $s \to t$ or $s \to_n t$ is derivable from S, then also $s' \to t'$ or respectively $s' \to_n t'$ is derivable from S. We have that $r \to_0 r$ is derivable from any S. If $r \to_m s$ and $s \to_n t$ are derivable from S, then $r \to_{m+n} t$ also. If $r \to_n s$ is derivable from S, it also happens for any greater n. If $r \to_{m+n} t$ is derivable from S, then there is an s such that $r \to_m s$ and $s \to_n t$ are derivable. Furthermore, if $r \to_n t$ is derivable, then there are s_0, s_1, $\ldots s_n$ such that each $s_i \to_1 s_{i+1}$ is derivable, s_0 is r and s_n is t.

§28

Bracket notation $r[r_1, \ldots, r_k]$ and TRS

We can say that $r \to_1 s$ is derivable from S if and only if we can obtain s by substituting some occurrences of open subterms r_i of r by s_i, where each $r_i = s_i$ is an instance of an equation of S. And that $r \to_n s$ or $r \to s$ is derivable if we can do this recursively, n times in the first case. When we deal with derivation of these objects, we say that S acts as a *term reduction system* [9]. The expression $r[r_1, \ldots, r_k]$ denotes an open term t, each open term r_i in this expression points to some occurrences of an open subterm of t formally equal to r_i, the subterms pointed by different r_i and r_j do not overlap. After substituting the open subterms pointed by each r_i by copies of s_i we get the open term $r[s_1, \ldots, s_k]$. Hence, we have $r[r_1, \ldots, r_k] \to_1 r[s_1, \ldots, s_k]$ is derivable from S if each $r_i = s_i$ is an instance of an equation of S, every $p \to_1 q$ derivable from S has this form.

2.3 Solvability, Consistence and Confluence

§29

Solvability and consistency

An equation $s_1 = s_2$ is **reducible** with a system T if there is an open term r such that $s_1 \to r$ and $s_2 \to r$ are derivable from S, it is (m, n)**-reducible** with T if there is an open term r such that $s_1 \to_m r$ and $s_2 \to_n r$ are derivable from S. The set of equations that are $(1, 1)$-reducible with a system T contains T and is closed under the instantiation, the ref, sym and cons schemata: this plays an important rôle.

An equation is called **decomposable** if its two parts have equal symbols as operators, non of these being an unknown, and an equal number of arguments. **Undecomposable** equations are divided in *directed* equations and *contradictions*. A **directed** equation is one having an unknown as operator in one of its two parts: it is directed **to the right** if its left part has an unknown as operator, it is directed **to the left** if its right part has an unknown as operator, it is a **bidirectional** equation if both parts have unknowns as operators, namely, if it is directed to the right and to the left at the same time. Undecomposable, **non directed** equations are **contradictions**, they are divided into *operators clashes* and *arguments clashes*. An **operators clash** is an equation whose two parts have formally different symbols as operators, non of these being an unknown. An **arguments clash** is an equation whose two parts have formally equal symbols as operators, non of them being an unknown, but a different number of arguments.

A system is **undecomposable** if all its equations are undecomposable. For a system S we define the equivalent, undecomposable system $\mathrm{dec}(S)$ recursively: we only need to recursively replace every decomposable equation of the form $fr_1 \cdots r_n = fs_1 \cdots s_n$ in S by the smaller, perhaps decomposable, equations $r_i = s_i$, this also means to delete the equation when $n = 0$. The systems S and $\mathrm{dec}(S)$ are equivalent: the equations of $\mathrm{dec}(S)$ are obtained from the ones of S with the dec and sep schemata, the equations of S are obtained from the ones of $\mathrm{dec}(S)$ with ref and cons.

We are specially interested on systems containing only equations directed to the right to be seen as term reduction systems *for substituting unknowns*. If $p \to_n r$ is derivable from such a system and if the operator f of p is not an unknown, then r has the same operator and number of arguments as p; furthermore, if this p is of the form $(fp_1 \cdots p_m)$, then r is of the form $(fr_1 \cdots r_m)$ so that each $p_i \rightarrow_n r_i$ is derivable from the system. A system is **directed to the right** if each of its equations is either a contradiction or directed to the right. An **orientation to the right** of an undecomposable system is obtained by replacing some of its equations $s = t$ by $t = s$ in order to obtain a system directed to the right. Since bidirectional equations and contradictions can be replaced or not, an orientation to the right is not necessarily unique. The equivalence of the new system to the original is provided by the open m-schema sym. If $s = t$ is in a system S and T is an orientation to the right of dec(S), then $s = t$ is $(1,1)$-reducible with T: the set T^* of the equations that are $(1,1)$-reducible with T contains T, is closed by sym and hence contains dec(S), is closed by ref and cons and hence contains S.

A system is **inconsistent** if a contradiction can be derived from it, **consistent** if no contradiction can be derived from it. A system T, seen as a term rewriting system, is a **solution** of S, if all equations of T are directed to the right and each equation $s = t$ derivable from S is reducible with T. Note that $T \vdash S$ holds: we call T a **most general** solution if also $S \vdash T$ holds. One can prove that S is consistent if and only if there is a solution T of it. If S is consistent, then the set T of all equations directed to the right derivable from S is a most general solution: if $s = t$ is derivable from S, then any orientation to the right of dec($\{s = t\}$) is, because it does not contain a contradiction, contained in T, and the $(1,1)$-reducibility of $s = t$ with T was proved in the paragraph above. If $s \rightarrow r$ and $t \rightarrow r$ are derivable from a solution T of S, then either s or t has an unknown as operator or both, s and t, have the same operator and the same number of arguments as r; in both cases $s = t$ cannot be a contradiction, and this is the case for any equation derivable from S.

Given a system S, we can enumerate the equations derivable from it, stop enumerating when a contradiction is found, or continue enumerating and collecting the equations directed to the right. Our main problem is to decide if a contradiction is derivable or to describe a most general solution without enumerating ad infinitum in cases where it is possible. If we insist on enumerating, then at least not all equations that are derivable, but a subset behaving in the same form: leading to a contradiction when there is one or building an enumeration of an eventual most general solution, so that the enumeration be easier, so that the eventual decision on the consistency be more probable, for example by stopping when there is nothing more to enumerate, so that the eventual description of the most general solution be simpler. Such an enumeration may be a point of departure for a more sofisticated *solving strategy*.

§30

Confluence

A system T is **confluent** if for every pair $t \rightarrow s_1$ and $t \rightarrow s_2$ derivable from T there is an open term r such that also $s_1 \rightarrow r$ and $s_2 \rightarrow r$ are derivable from T.

The set S of equations reducible with a given confluent system T whose equations are directed to the right contains the equations of the system T and is closed under all open m-schemata; hence, an equation is reducible with such a T if and only if it is derivable from it: The set S obviously contains T and is closed by instantiation and the ref, sym and cons schemata, is closed by trans because of the confluence, is closed by dec and sym because the equations of T are directed to the right. Of course, the use of the word "set" should be in this proof, as everywhere in this article, correctly interpreted: we are giving a hint how to recursively find the corresponding r such that $s \to r$ and $t \to r$ are derivable from T for every equation $s = t$ in a derivation tree.

If we allow that T contain contradictions, namely, if we weaken the hypothesis that all equations of T are directed to the right and only suppose that T is oriented to the right, then the process of of finding the r for each equation of a derivation tree may fail at the underequation of a rule of the dec or sep schema: in the derivation of $s_i \to r$ from T, where s_i is one of the parts of the overequation and r the corresponding r, may occur a contradiction in T as firstequation. And this must be the case when there is a contradiction in the derivation. We conclude: If a contradiction is derivable from a confluent system oriented to the right, then a contradiction can be found in the system.

For *solving* a system, we could try to enumerate a confluent system oriented to the right equivalent to it: If the system is inconsistent, we get in this way sooner or later a contradiction; if the system is consistent, we are enumerating a most general solution. The problems of deciding on consistency and of expressing a most general solution remain.

§31

(m, n, i, j)-confluence

We continue with the *parallelogram* law for TRS. A system T is (m, n, i, j)**-confluent** if for every pair $t \to_m p$ and $t \to_n q$ derivable from T there is an open term r such that also $p \to_i r$ and $q \to_j r$ are derivable from T. Obviously (m, n, i, j)-confluence and (n, m, j, i)-confluence are equivalent. Every T is obviously $(0, n, n, 0)$- and $(m, 0, 0, m)$-confluent for every n and m. An $(1, 1, 1, 1)$-confluent system is (m, n, n, m)-confluent for every m and n, and hence confluent. For proving this, we see first that $(1, n, n, 1)$- and (m, n, n, m)-confluence implies $(m + 1, n, n, m + 1)$-confluence: if $t \to_{m+1} p$ and $t \to_n q$ are derivable, then there is a p' such that $t \to_m p'$ and $p' \to_1 p$ are derivable, and due to the (m, n, n, m)-confluence an open term r' such that $p' \to_n r'$ and $q \to_m r'$ are derivable, and due to the $(1, n, n, 1)$-confluence an r such that $p \to_n r$ and $r' \to_1 r$ are derivable; hence, $p \to_n r$ and $q \to_{m+1} r$ are derivable. By induction on m, we have that $(1, n, n, 1)$-confluence implies (m, n, n, m)-confluence for each m. Symmetrically to this, $(m, 1, 1, m)$-confluence implies (m, n, n, m)-confluence for each n. From these two last results follows our main result.

A system S is $(1, 1, 1, 1)$-confluent if and only if for each instance $t = p$ of an equation in S and each $t \to_1 q$ derivable from S the equation $q = p$ is $(1, 1)$-reducible with S. This condition is trivially fulfilled when q is formally equal to p or to t. This statement is proved by induction on the size of two given derivations built of ref and cons-rules and whose endequations are $t \to_1 p$ and $t \to_1 q$: if one of the endequations is an instance of an equation in S, then we find the desired r by using the hypothesis; if one of these endequations is the underequation of a ref-rule, then the desired r is the right part of the other endequation; if both are underequations of cons-rules, then the result follows trivially by the inductive hypothesis.

Our first solving strategy is the use of a variant of Knuth-Bendix' algorithm to enumerate a $(1,1,1,1)$-confluent system oriented to the right. The above result plays the essential rôle, but for treating universal quantification we need something like a *lifting lemma*, we need some remarks on classical term unification.

2.4 Knuth-Bendix' Algorithm for Solving Equations

Subsection 2.5 is independent of this subsection and could serve as an introduction to it.

§32

A note on term unification

We introduce now, independent of our treatment of term unification in 2.5, the result we need for our algorithm for solving equations systems. We need term unification for finding appropriate instances, for substituting variables with open terms: variables play the rôle of unknowns in 2.5, term unification is considered outside its logical context introduced in 2.5.

With t^α we denote here the open term obtained by substituting the variables of the open term t by the open terms given by the *substitution* α that associates open terms to variables. An instance of an equation $s = q$ is always of the form $s^\alpha = q^\alpha$. Substituting the variables in an open term $t[r_1, \ldots, r_n]$ yields the new open term $t^\alpha[r_1^\alpha, \ldots, r_n^\alpha]$, where r_i^α points to its open subterms that are in the places originally pointed by r_i; hence, we can substitute each r_i^α in this open term by an open term q_i' for obtaining the open term $t^\alpha[q_1', \ldots, q_n']$. If b^α is of the form $\bar{t}[\bar{r}_1, \ldots, \bar{r}_m]$, then each $\bar{r}_i$ points to subterms of b^α of the form r^α being in the places of possibly formally different subterms r of b; hence, we can write b in the form $t[r_1, \cdots, r_n]$, where n is not necessarily equal to m.

A **unifier** for reducing an instance of the open term $t[r_1, \ldots, r_n]$ with the equations $s_1 = q_1$, …, $s_n = q_n$ is a list $(\alpha, \alpha_1, \ldots, \alpha_n)$ of substitutions like above such that r_i^α and $s_i^{\alpha_i}$ be formally equal, the **reduction** of the instance $t^\alpha[r_1^\alpha, \ldots, r_n^\alpha]$ of this open term with these equations and unifier is the open term $t^\alpha[q_1^{\alpha_1}, \ldots, q_n^{\alpha_n}]$. A **most general** unifier is a unifier such that all other unifiers are of the form $(\alpha\gamma, \alpha_1\gamma, \ldots, \alpha_n\gamma)$. The results in [11] yields an algorithm for deciding if there is an unifier and for giving a most general unifier when there is an unifier: we consider a unifier to be a list of substitutions because we see occurrences of a variable in the context of different equations as different variables that can be substituted by formally different open terms; the domains of definition of α, α_1, …, α_n are to be seen as pairwise disjoint, then we can see the list $(\alpha, \alpha_1, \ldots, \alpha_n)$ as a substitution acting on the union of the domains of definition; a renaming of the variables in the equations so that variables in different equations and in the original open term be pairwise disjoint must be done before applying the traditional unification algorithm, v^α or v^{α_i} is different from v^{α_j} if v does not appear in $t[r_1, \cdots, r_n]$ or respectively $s_i = q_i$ because the substitutions contain the renaming.

If $b^\beta \to_1 \bar{q}$ is derivable from S, then b can be expressed in the form $t[r_1, \ldots, r_n]$ and $\bar{q}$ in the form $t^\beta[\bar{q}_1, \ldots, \bar{q}_n]$, so that each $r_i^\beta = \bar{q}_i$ is an instance $s_i^{\beta_i} = q_i^{\beta_i}$ of an equation $s_i = q_i$ in S. In this case, $(\beta, \beta_1, \ldots, \beta_n)$ is a unifier for reducing the instance $t^\beta[r_1^\beta, \ldots, r_n^\beta]$ of $t[r_1, \ldots, r_n]$, namely b^β, into $t^\beta[\bar{q}_1, \ldots, \bar{q}_n]$, namely $\bar{q}$, with the $s_i = q_i$

in S: each $\bar{q}_i$ is formally equal to $q_i^{\beta_i}$. A most general unifier $(\alpha, \alpha_1, \ldots, \alpha_n)$ such that $(\alpha\gamma, \alpha_1\gamma, \ldots, \alpha_n\gamma)$ be equal to $(\beta, \beta_1, \ldots, \beta_n)$ yields a reduction q of the instance b^α of b. Now, the original equation $b^\beta \to_1 \bar{q}$ is the instance $b^{\alpha\gamma} \to_1 q^\gamma$ of $b^\alpha \to_1 q$, and this last equation is also derivable from S.

§33

cp-resolution

A **cp-resolvent** of a list of equations $t[r_1, \ldots, r_n] = p$, $s_1 = q_1$, $\ldots$, $s_n = q_n$ is an orientation to the right of $\mathrm{dec}(\{t^\alpha[q_1^{\alpha_1}, \ldots, q_n^{\alpha_n}] = p^\alpha\})$, where $(\alpha, \alpha_1, \ldots, \alpha_n)$ is a most general unifier for reducing an instance of $t[r_1, \cdots, r_n]$ with $s_1 = q_1$, $\ldots$, $s_n = q_n$. The existence of the most general unifier is a condition for the existence of a cp-resolvent. If each equation in the list is derivable from a system, then also each equation in a cp-resolvent.

A system S is **cp-closed** if it is oriented to the right and for each list of equations having a cp-resolvent there is a cp-resolvent each of its equations is $(1,1)$-reducible with S. A cp-closed system is $(1,1,1,1)$-confluent and hence also confluent: according to the last result in §31, it is enough to see that $\bar{q} = p^\beta$ is $(1,1)$-reducible for each instance $b^\beta = p^\beta$ of an equation $b = p$ in S and each $b^\beta \to_1 \bar{q}$ derivable from S; following the last paragraph in §32, b is of the form $t[r_1, \ldots, r_n]$ and there is an instance b^α of it that can be reduced with some equations $s_i = q_i$ of S and with a most general unifier $(\alpha, \alpha_1, \ldots, \alpha_n)$ into a q, so that $\bar{q}$ be an instance q^γ of q and β be $\alpha\gamma$; a corresponding cp-resolvent of the equations $t[r_1, \ldots, r_n] = p$, $s_1 = q_1, \ldots s_n = q_n$ of S is an orientation to the right of $\mathrm{dec}(q = p^\alpha)$; since there is a cp-resolvent all of its equations are $(1,1)$-reducible, $q = p^\alpha$ and hence its instance $\bar{q} = p^\beta$ are also $(1,1)$-reducible.

A cp-closed system is confluent and oriented to the right: it contains a contradiction if it is inconsistent, it is a (most general) solution of itself and of any system equivalent to it if it is consistent.

Given a system S, one can build a cp-closed system S^* equivalent to S by recursively adding to an orientation to the right of $\mathrm{dec}(S)$ the equations of cp-resolvents of equations in it. We call this process **cp-completion**, during it we must find a contradiction if S is inconsistent, or enumerate the equations of a most general solution of S if S is consistent.

2.5 Term Unification

Since we want that solution of equations systems play the rôle of term unification in logic programming, we introduce term unification in the context of the concepts we introduced for equations systems. This subsection is an appendix that can be skipped.

§34

Substitutions

A substitution α is a function that associates a term X^α to each unknown X from a finite set sup_α. For each open term t, we define recursively the open term t^α. If t is an unknown in sup_α, then t^α is given by the definition of α; if t is other 0-ary symbol, t^α

coincides with t; if t is of the form $(t_1 t_2)$, then t^α is $(t_1^\alpha t_2^\alpha)$. Two substitutions α and β are considered the same if X^α coincides with X^β for each unknown X, or equivalently, with each open term X, even if $\sup_\alpha$ and $\sup_\beta$ does not coincide. For open terms s and t, or for substitutions α and β, we say that the identity $s \equiv t$ holds, or respectively that the identity $\alpha \equiv \beta$ holds, if right and left part of the identity coincide.

The identity $t^\alpha \equiv t$ holds if and only if for every unknown X in t the identity $X^\alpha \equiv X$ holds. We say that α **moves** an open term t, if t^α is different from t: it moves t to t^α. All unknowns moved by α are in $\sup_\alpha$.

The **unity** ι is the substitution for which X^ι coincides with X for each unknown X, and hence also for each open term X. The associative product $\alpha\beta$ of two substitutions α and β is the only substitution for which $X^{\alpha\beta}$ coincides with $(X^\alpha)^\beta$ for each unknown X, and hence, also for each open term X. The set of all substitutions build together with this product and with ι a *monoid*.

§35

Idempotent substitutions, ordering, equivalence

A substitution α is called **idempotent** if $\alpha\alpha \equiv \alpha$ holds. The substitution α is idempotent if and only if for every X moved by α the unknowns in X^α are not moved by α. If we call the unknowns moved by α *dependent* and the rest *independent*, then we can say that α express dependent unknowns in terms of independent ones.

For two substitutions α and β, let the relation $\alpha < \beta$ hold if and only if $\alpha \equiv \beta\alpha$ holds. The relation $<$ is transitive: from $\alpha \equiv \beta\alpha$ and $\beta \equiv \gamma\beta$ the identity $\alpha \equiv \beta\alpha \equiv (\gamma\beta)\alpha \equiv \gamma(\beta\alpha) \equiv \gamma\alpha$ follows. The relation $\alpha < \alpha$ holds if and only if α is idempotent. Let the relation $\alpha \sim \beta$ hold if and only if $\alpha < \beta$ and $\beta < \alpha$ holds. The relation $\sim$ restricted to idempotent substitutions is an equivalence relation. If $\alpha \sim \beta$ holds, then α and β are idempotent: $\alpha\alpha \equiv \alpha(\beta\alpha) \equiv (\alpha\beta)\alpha \equiv \beta\alpha \equiv \alpha$, the proof for β is symmetrical.

If $\alpha \sim \beta$ holds, then there is a substitution γ such that $\gamma\gamma \equiv \iota$, $\alpha\gamma \equiv \beta$ and $\beta\gamma \equiv \alpha$ holds: α and β are equal up to a renaming. For proving this, we note first that X^α is an unknown if and only if X^β is an unknown: if X^α is an unknown and X^β of the form $(t_1 t_2)$, then $X^\alpha \equiv X^{\beta\alpha} \equiv (t_1^\alpha t_2^\alpha)$ is a contradiction; the proof for the other direction is symmetrical. Let Γ be the set of X such that X^α, or equivalently such that X^β, is an unknown. This Γ contains all unknowns outside $\sup_\alpha \cap \sup_\beta$, perhaps contains some in it. Let Γ^α and Γ^β be the sets of unknowns obtained by substituting each X in Γ by X^α or respectively by X^β. We can restrict β to a function β' from Γ^α to Γ^β and α to a function α' from Γ^β to Γ^α, so that α' is the inverse function of β': this follows from the identities $(X^\alpha)^\beta \equiv X^\beta$ and $(X^\beta)^\alpha \equiv X^\alpha$. The functions β' and α' restricted to the set $\Gamma^\alpha \cap \Gamma^\beta$ coincide with the identity: from $X^\alpha \equiv Y^\beta$ follows $X^{\alpha\beta} \equiv Y^{\beta\beta}$, due to the definition of $\sim$ and to the idempotence of β, also $X^\beta \equiv Y^\beta$, and with the first equation $X^\beta \equiv X^\alpha$, namely, the restriction of β' coincide with ι; the proof for α' is symmetrical. We define the function γ' by joining the graphs of β' and α', and the substitution γ by restricting γ' to the finite set $(\Gamma^\alpha \cup \Gamma^\beta) \cap (\sup_\alpha \cup \sup_\beta)$; the natural extension of γ to unknowns in the domain $\Gamma^\alpha \cup \Gamma^\beta$ of γ', but outside $\sup_\gamma$ coincide with γ': the function γ' coincide with ι in the set $\Gamma^\alpha \cap \Gamma^\beta$ containing all elements outside the finite set $\sup_\alpha \cup \sup_\beta$. The identity $\gamma\gamma \equiv \iota$ is obvious from the definition of γ.

Since α is idempotent, no unknown in X^α for an arbitrary unknown X is moved by α, they are hence in Γ and in Γ^α, and γ assign to them the same unknown as β does: $(X^\alpha)^\gamma \equiv (X^\alpha)^\beta \equiv X^{\alpha\beta} \equiv X^\beta$. From this the identity $\alpha\gamma \equiv \beta$ follows, the proof of the identity $\beta\gamma \equiv \alpha$ is symmetrical.

§36

Substitutions as systems

To each substitution α we associate an equations system S_α whose equations are the ones of the form $X = X^\alpha$ for which the unknown X does not coincide with X^α. A system S whose equations are directed to the right, contain no variable and have pairwise different unknowns in their left parts is of the form S_α for a substitution α. The identity $\alpha \equiv \beta$ holds if and only if S_α and S_β contain the same equations.

The system S_α is oriented to the right and, according to §31, is $(1,1,1,1)$-confluent, hence also confluent. Since all its equations are directed to the right, it contains no contradiction as defined in §29, and hence no such is derivable, but later we will introduce a new kind of contradictions that may be derivable from an S_α whose α is not idempotent.

The identity $S_\alpha \vdash t \to_1 t^\alpha$ holds for a substitution α and an open term t. This follows by induction. It is obvious when t is a 0-ary symbol. From $S_\alpha \vdash t_1 \to_1 t_1^\alpha$, $S_\alpha \vdash t_2 \to_1 t_2^\alpha$, the cons schema and the identity $(t_1 t_2)^\alpha \equiv (t_1^\alpha t_2^\alpha)$ follows $S_\alpha \vdash (t_1 t_2) \to_1 (t_1 t_2)^\alpha$.

§37

Unifiers

A substitution α **unifies** the equation $s = t$ if the identity $s^\alpha \equiv t^\alpha$ holds. It unifies a system S if it unifies each of its equations. If α unifies S and $S \vdash s = t$ holds, then α unifies $s = t$: if a substitution unifies the overequations of an open m-rule, then also the underequation. Two equivalent systems have the same unifiers.

If α unifies an equation or system, then also $\alpha\gamma$ for any γ. A substitution β unifies S_α if and only if $\beta \equiv \alpha\beta$, namely $\beta < \alpha$, holds: this is a direct consequence of the definitions. In particular, α unifies S_α if and only if α is idempotent. Joining the above remarks: a substitution unifies S_α, where α is idempotent, if and only if it is of the form $\alpha\gamma$.

If α unifies $s = t$, then $S_\alpha \vdash s = t$ holds: this follows from the fact that $S_\alpha \vdash s \to_1 s^\alpha$ and $S_\alpha \vdash t \to_1 t^\alpha$ hold. If α is idempotent, then $S_\alpha \vdash s = t$ if and only if α unifies $s = t$.

The task of the *unification algorithm* is to describe the unifiers β of a system S, where the β and S must satisfy some *type constraints*. We consider the task of transforming S into an equivalent system S_α, where α is idempotent. Then, the unifiers of S are the substitutions of the form $\alpha\gamma$. This idempotent α is *unique up to renaming*: $S_{\alpha_1} \equiv S_{\alpha_2}$ implies $\alpha_1 \sim \alpha_2$. We prove that this task is equivalent to the original one: the transformation is possible if and only if there is a unifier, and this is decidable.

§38

Infinite loops, contradictions

No substitution unifies a contradiction: if $s = t$ is an operators or an arguments clash, then $s^\alpha = t^\alpha$ is a similar contradiction, its two parts cannot coincide.

An **infinite loop** is a directed equation of the form $X = t$ or $t = X$, in which t is an open term of the form $(t_1 t_2)$ containing X. No substitution unifies such an equation: t^α contains X^α as subterm and other symbols, they cannot coincide. Slightly diverging from the definitions in §29, we consider **in this section** infinite loops also as **contradictions**. As in §29, a system is consistent if and only if no contradiction, including infinite loops, is derivable.

Summarizing, no substitution unifies an inconsistent system. No substitution unifies the consistent system containing the equation $(Xa) = a$, where X is an unknown and a a ground symbol. We prove that, under certain conditions, a consistent system S can be transformed into an equivalent system S_α, where α is idempotent and hence a unifier of S_α and S.

§39

Forbidding applications of unknowns, T-terms

An **open T-term** is constructed recursively: an open term consisting of only one symbol is an open T-term; an open term of the form $(t_1 t_2)$, where t_1 is not an unknown and both, t_1 and t_2, are open T-terms, is also an open T-term. A **T-term** is an open T-term not containing variables. A **T-equation** is one whose two parts are open T-terms. A **T-system** is a system whose equations are T-equations.

A substitution α is a **T-substitution** if X^α is a T-term for each unknown X, namely, if S_α is a T-system. If t is an open T-term, then also t^α. If α and β are T-substitutions, then also $\alpha\beta$.

Open terms were constructed with a signature, all symbols are 0-ary with exception of the 2-ary application operator. With a second signature assigning a second arity to each 0-ary symbol, one can construct open terms such that the number of arguments of them are given by the second arity of their operators. If the second arity of all unknowns is 0, then the open terms constructed are open T-terms: the T stays unproperly for "typed". Expressions constructed with a usual signature can be embedded in our system with two signatures.

§40

Directed T-equations, directed contradictions

Undecomposable T-equations are either a directed equation of the form $X = t$ or $t = X$, where t is an open T-term, or an operators clash or an arguments clash.

A **directed contradiction** is either an infinite loop or a directed T-equation containing a variable. From an equation of the last kind one can derive an operators clash. With sym and trans one can derive a T-equation $t[v] = t[w]$ from it and an instance of it, where $t[v]$ is the part of the equation containing the variable v and w a new variable. If $t[v]$ contains more than one symbol, then its operator is not an unknown, and with dec or sep we can obtain a smaller equation $t'[v] = t'[w]$ of the same form. With some applications of dec and sep, we obtain the contradiction $v = w$. We also speak about contradictions directed **to the right** or **left**. In this section, **contradictions** are either an operators clash, or an arguments clash, or a directed contradiction, or a loop.

§41
Unification algorithm as consistence checking

We consider now T-systems S whose equations are arranged in two lists: a list U of undecomposable T-equations, each being either a contradiction or directed to the right, and a list R of T-equations $X = t$ directed to the right so that t contains no variable and X appear neither in t nor in other equation of S. We denote this system S with (R, U), the list R stays for "resolved" and U for "unsolved". Eventually we treat R and U also as systems.

If U contains a contradiction, then (R, U) is inconsistent. If U contains only equations of the form $X = X$, then the system (R, U) is clearly equivalent to one of the form S_α, where α is idempotent, and hence consistent. For an arbitrary system S there is always a system of the form $(\emptyset, U)$ equivalent to it, U is built of an orientation to the right of dec(S). Given a system (R, U) and a directed T-equation $X = t$ from U, so that t contains neither X nor a variable, we can transform (R, U) into an equivalent system (R', U'): the equation $X = t$ is deleted from the second list, each occurrence of X (in the first or second list) is substituted by t, the equation $X = t$ is added to right end of the first list, each equation $s_1 = s_2$ in the second list is substituted by the equations of an orientation to the right of dec($\{s_1 = s_2\}$) (in some order). It is not difficult to see that (R', U') satisfy the required conditions; furthermore, the number of unknowns occurring in U' is less than the number of the ones occurring in U. The recursive application of such transformations to a system (R, U) with arbitrarily selected equations $X = t$ must terminate. If a system (R, U) cannot be reduced, then each equation of U is either of the form $X = X$ or a directed contradiction or another contradiction (clash): examining U we can decide if the system is consistent or not, if there is a unifier or not, if the system is equivalent to an S_α with an idempotent α or not.

Summarizing, we can decide whether a T-system S is consistent or not. In the first case, S can be transformed into an equivalent system S_α, the symbols of its equations being symbols from equations in S and α being idempotent. In the second case, we can derive a contradictory T-equation from S whose symbols are in the equations of S.

§42
Comb lemma

We call the following result the **comb lemma** for most general unifiers, it is necessary in the last six lines of [16], in §29, for proving the lifting lemma in classical logic programming. Let $X_0, X_1, \ldots, X_n \subseteq X$ be pairwise disjoint sets of unknowns satisfying $X = \bigcup_{i=0}^{n} X_i$; let $S_1, \ldots, S_n$ be systems, so that the equations of an S_i do not contain any unknown of X_j when $j \neq 0$ and $j \neq i$; let q be a symbol not contained in X, and α_i

an unifier of S_i for each i, so that q does not occur in an x^{α_i} when $x \in X_0$; then q does not appear in any x^α when x is in X_0 and α is a most general unifier for $\bigcup_{i=1}^n S_i$. We prove now this lemma. A system S is **solved** if it is of the form S_η with idempotent η. For every S_i let β_i be a unifier of S_i, so that S_{β_i} be equivalent to S_i and all symbols of S_{β_i} appear in S_i. Let $E_i := \{x = \varphi \in S_{\beta_i} : x \in X_i\}$, $K_i := \{x = \varphi \in S_{\beta_i} : x \in X_0\}$, $E := \bigcup_{i=1}^n E_i$, $K := \bigcup_{i=1}^n K_i$. For every equation $x = \varphi \in S_{\beta_i}$ we have, due to the choice of β_i and the hypotheses, that either $x \in X_0$ or $x \in X_i$ holds and that φ contains no $y \in X_j$ when $j \neq 0$ and $j \neq i$; hence, $S_{\beta_i} = E_i \cup K_i$ holds and E is a solved system; furthermore, $K \cup E$ is equivalent to $\bigcup_i S_i$. Since α unifies $\bigcup_i S_i$ and hence the system $K \cup E$ equivalent to it, one can find a unifier κ of K, such that S_κ be equivalent to K and contain only symbols from K; the system $S_\kappa \cup E^\kappa$ is equivalent to $S_\kappa \cup E$ and hence also to $K \cup E$ and to $\bigcup_i S_i$, it is a solved system and contain only symbols occurring in $\bigcup_i S_i$; let β be a substitution satisfying $S_\beta = S_\kappa \cup E^\kappa$. Since β_i is a most general unifier of S_i and α_i a solution of this same system, $\alpha_i = \beta_i \alpha_i$ holds; due to this equality and the hypotheses, q does not occur in x^{β_i} for x in X_0 and arbitrary i, also not in a K_i, and hence also not in K; since symbols occurring in S_κ also occur in K, the symbol q occurs in no $x^\beta \equiv x^\kappa$ with $x \in X_0$; since α is a most general unifier of $\bigcup_i S_i$ and β a unifier of the same system, the equality $\beta \equiv \alpha\beta$ holds, and hence q cannot occur in an x^α with $x \in X_0$.

3 Logic Programming for Stating Equations

This section is devoted to restate the task of logic programming (subsection 3.2), so that the concept of solution of equations systems introduced in section 2 play the rôle of term unification, and to prove the lifting lemma (subsection 3.4) that means the completenes of an indeterministic search algorithm. Given a sequent $\Sigma \vdash \varphi$, the old task of logic programming was to find a substitution α for the unknowns in the sequent, so that $\Sigma^\alpha \vdash \varphi^\alpha$ be mj-derivable. We can say that the new task is to find a consistent system of equations α, so that $\Sigma \cup \alpha \vdash \varphi$ be mj-derivable. Well, some technical details should be make more precise in this definition. Having a substitution α gives us a finite evaluation procedure of each unknown, with an equations system we do not have such an evaluation procedure: this makes the proof of the lifting lemma much more complicated than in classical logic programming.

3.1 Logic with Unknowns and Equality

§43

Open terms and formulae, sequents with infinite antecedents

We consider in this section open terms and formulae built on a language L for the predicate logic. This language L contains a 2-ary predicate symbol $=$, its functional symbols are a 2-ary application operator and 0-ary functional symbols divided in unknowns and ground symbols, as auxiliary 0-ary symbols we have infinite many for variables and for free variables.

An **equation** in L is an atomic open formula with the predicate $=$, it may contain symbols for free variables. A **system of equations** in L is defined as in 2.1, with equality axioms as in 2.1, but its equations can now contain symbols for free variables. For applying the results of section 2.1 to these systems, symbols for free variables are to be seen as unknowns: axioms dec(f) are for ground terms f and the schema sep for variables, not for free variables. Actually, symbols for free variables are to be used as proper variables in the EB- and AE-rules defined in subsection 1.1, hence as auxiliary terms for the construction of mj-compositions, they are the free variables in Gentzen [4] or the parameters in Prawitz [15], they are unwelcome in the endsequent, they are unwelcome in equations systems; but for stating our results, we need to consider the possibility that they appear in equations systems. The equations we deal with in this section are succedents of firstsequents of mj-compositions, some of the symbols for free variables in them correspond to proper variables of AE-rules, these symbols are seen as arbitrary constants, they are substituted later by variables for building systems with the equations, only the remaining symbols for free variables are seen as unknowns.

We consider sequents $\Sigma \vdash \varphi$ with infinite antecedent Σ, such that there are infinitely many symbols for free variables not occurring in Σ. To be more precisely, sequents of the form $S \cup P \vdash \varphi$, where S is a system of equations (and hence not containing free variables) arranged as a list in some way and P a (finite) list of formulae. NJ- and mj-rules are defined exactly as before; furthermore, $\Sigma \vdash \varphi$ is derivable if and only if a sequent $\Sigma' \vdash \varphi$ is derivable, where Σ' is a finite list obtained by deleting formulae from Σ.

§44

Compatibility axioms

We define for an n-ary predicate symbol P the formula comp(P), called a **compatibility** axiom, by

$$\text{comp}(P):$$
$$\forall u_1 \cdots \forall u_n \forall v_1 \cdots \forall v_n u_1 = v_1 \& \cdots \& u_n = v_n \supset (P(u_1, \ldots, u_n) \supset P(v_1, \ldots, v_n)),$$

where & denotes the nested implications that can be substituted by conjunction for obtaining an equivalent formula. Note that comp(=) is a consequence of sym and trans. With comp_L we denote the list of compatibility axioms comp(P) for each predicate symbol P diferent from $=$ of the language L we are considering.

3.2 New Statement of Logic Programming

§45

DE- and GE-Formulae, PE-sequents

We consider here mj-derivability of P-sequents $S \cup \Sigma \vdash \varphi$, where S is an equations system and Σ contains comp_L. We want to restrict these P-sequents, so that the derivability of $S \cup \Sigma \vdash \varphi$ implies the derivability of $S \vdash \varphi$ when φ is the closure of an equation.

An **open AE-formula** is an open A-formula not containing the predicate symbol $=$. An **open E-formula** is an equation. By simultaneous recursion we define **open DE- and GE-formulae**:

$$\mathrm{DE} := \mathrm{AE}|\forall v\mathrm{DE}|\mathrm{DE}_1 \wedge \mathrm{DE}_2|\mathrm{GE} \supset \mathrm{DE},$$
$$\mathrm{GE} := \mathrm{E}|\mathrm{AE}|\exists v\mathrm{GE}|\forall v\mathrm{GE}|\mathrm{GE}_1 \vee \mathrm{GE}_2|\mathrm{GE}_1 \wedge \mathrm{GE}_2|\mathrm{DE} \supset \mathrm{GE}.$$

By induction we can prove that they are open D- and respectively G-formulae. AE-, E-, DE- and GE-formulae are open AE-, E-, DE- and GE-formulae whose variables are bound with quantifiers. All formulae in comp_L are DE-formulae (but not $\mathrm{comp}(=)$). A **PE-sequent** is one of the form $\mathrm{comp}_L \cup P \vdash \varphi$, where P is a list of DE-formulae called **proper assumptions** of the PE-sequent and φ a GE-formula. If the undersequent of a mj-rule is a PE-sequent, then also its oversequents are so. Since no D-sequence beginning with a DE-formula ends with an E-formula, a PE-sequent $\Sigma \vdash \varepsilon$ whose succedent ε is the closure of an equation cannot be the undersequent of an m-rule, such sequents are not mj-derivable.

§46

Potential mj-derivations, PS-sequents, E-decomposition, resolvents, actualizations

A **potential** mj-derivation Ξ is an mj-composition tree whose endsequent $\Sigma \vdash \varphi$ is a PE-sequent and such that the succedents of its firstsequents are equations ε_1, $\ldots$, ε_n, called the **conditions** of Ξ. In a condition ε_i of Ξ may occur a (symbol for free variable formally equal to a) proper variable of Ξ (as defined in §10), let ε_i' be an equation obtained by substituting these proper variables by variables, formally different free variables by formally different variables: the system S whose equations ε_1', $\ldots$, ε_n' are obtained in this way from the conditions ε_1, $\ldots$, ε_n of Ξ is the **most general resolvent** of Ξ, it is unique up to renaming of variables. By appending S to the antecedents of the sequents in Ξ and by putting an m-rule over each firstsequent we can build an mj-derivation of $S \cup \Sigma \vdash \varphi$.

A **PS-sequent** is one of the form $S \cup \Sigma \vdash \varphi$, where S is an equations system, **its** equations system, and $\Sigma \vdash \varphi$ a PE-sequent, whose proper assumptions are also the ones of the PS-sequent. If the undersequent of an mj-rule is a PS-sequent $S \cup \Sigma \vdash \varphi$, then also its oversequents are PS-sequents with the same equations system S.

If the undersequent of an mj-rule is a PS-sequent whose succedent is the closure of an equation, then its oversequents are of the same form and have the same antecedent (an equation without variables is the closure of itself). In particular, the first formula in Δ of such a rule $m(S \cup \Sigma, \Delta)$ must be from the equations system S of the undersequent, because a D-sequent beginning with an DE-formula never ends with an E-formula. An mj-derivation Ξ of a PS-sequent $S \cup \Sigma \vdash \varepsilon$, where ε is the closure of an equation, can be transformed into an mj-derivation of $S \vdash \varepsilon$ by deleting Σ and leaving the equations system S in every PS-sequent of the derivation.

An mj-derivation Ξ of a PS-sequent $S \cup \Sigma \vdash \varphi$ can be transformed into a potential mj-derivation Ξ' of the PE-sequent $\Sigma \vdash \varphi$ whose most general resolvent is more general than the equations system S, namely, so that ε' is derivable from S if ε' is in the most general resolvent of Ξ'. This Ξ' is obtained by cutting off every branch of Ξ from the point at which an equation ε is found in the succedent, these equations ε will be the conditions of Ξ', and then by deleting the equations system S of each PS-sequent in the tree: it remains an mj-composition tree because the D-sequence of an m-rule the succedent of its undersequent is not an equation cannot begin with an equation of S. From the above paragraph, the subtree of Ξ whose endsequent $S \cup \Sigma' \vdash \varepsilon$ has a condition ε of Ξ' as succedent, namely, the subtree of Ξ built with the branches over $S \cup \Sigma' \vdash \varepsilon$ cut off from Ξ, can be converted into an mj-derivation of $S \vdash \varepsilon$; since S contains no proper variable of Ξ, and hence also no one of Ξ', the equation ε' of the resolvent of Ξ' obtained from ε by substituting the proper variables of Ξ' occurring in ε by variables is also derivable from S. We call the potential mj-derivation Ξ' together with the derivations of its firstsequents from S the **E-decomposition** of Ξ.

A **resolvent** of a potential derivation Ξ of a PE-sequent $\Sigma \vdash \varphi$ is an equations system S in which no proper variable of Ξ occur, so that the succedent of every firstsequent of Ξ is derivable from S, namely, so that the most general resolvent of Ξ be more general than S. By appending S to the antecedents of the sequents in Ξ and by appending some mj-derivations over each firstsequent of Ξ one gets an mj-derivation Ξ' of the PS-sequent $S \cup \Sigma \vdash \varphi$, called an **actualization** of Ξ by S. By E-decomposition of Ξ' one recovers Ξ and gets derivations of each equation in the most general resolvent of Ξ from S.

§47

The new task of logic programming

Given a PE-sequent $\Sigma \vdash \varphi$, we want to find a consistent system S, perhaps satisfying some additional conditions, so that the PS-sequent $S \cup \Sigma \vdash \varphi$ is mj-derivable. As in classical logic programming [10], we want a *systematic search strategy* (like the tree search with backtracking in Prolog) and, if an S is found, to take the *most general* S found in the same way (lifting).

From §46, we know that such an S is the resolvent of a potential derivation, and the most general resolvent of this potential derivation describes in some way all such S. We need only to *search* potential derivations of the PE-sequent $\Sigma \vdash \varphi$, but we have the problem of *selecting* the auxiliary terms. As in classical logic programming, we want to select unknowns representing auxiliary terms to be searched for, and to be sure that solutions found in this way lead through *specification* of these unknowns to every possible solution. This is the matter of the two following subsections.

3.3 Displacements

§48

S-displacements of open terms

Let S be a system of equations. We say that an open term t' is an S-**displacement** of the open term t, if $t = t'$ is an instance of an equation in S such that each variable occurring in t' also occurs in t. Hence, if t is a term, then also t'. Obviously, $S \vdash t \rightarrow_1 t'$ holds.

§49

S-displacements of open formulae

Let $r = r'$ be an equation from S, let $s = s'$ be an instance of $r = r'$ obtained by substituting each variable in r' not occurring in r by a term (without variables), let v_1, $\ldots v_n$ be some (perhaps all) variables in $s = s'$, for each v_i let t_i be a term and t'_i an S-displacement of it, let t be the open term obtained by substituting all occurrences of each variable v_i in s by t_i and t' the open term obtained by substituting all occurrences of each v_i in s' by t'_i. It is easy to see that $S \vdash t \rightarrow_2 t'$ holds. We say that the open term t' is a **second order S-displacement** of the open term t.

An **S-displacement** of an open atomic formula $R(t_1, \ldots, t_n)$ is an open atomic formula $R(t'_1, \ldots, t'_n)$, so that each t'_i be a second order S-displacement of t_i. The **S-displacement** of an arbitrary open formula is defined recursively: if φ' is an S-displacement of φ and ψ' an S-displacement of ψ, then $\varphi' \supset \psi'$ is an S-displacement of $\varphi \supset \psi$, $\varphi' \wedge \psi'$ an S-displacement of $\varphi \wedge \psi$, $\varphi' \vee \psi'$ an S-displacement of $\varphi \vee \psi$, $\forall v \varphi'$ an S-displacement of $\forall v \varphi$ and $\exists v \varphi'$ an S-displacement of $\exists v \varphi$.

If φ' is an S-displacement of φ and a variable v occurs in φ' not bound by a quantifier, then it also occurs in φ not bound by a quantifier; hence, φ' is a formula if φ is a formula: this is the case for second order S-displacements of open terms and for S-displacements of open atomic formulae, the general case is seen by induction. In an analogous way the following is proved: if φ' is an S-displacement of the open formula φ, if t' is an S-displacement of the term t, if ψ' is obtained by substituting each non-bound occurrence of a variable v in φ' by t' and ψ by substituting each non-bound occurrence of v in φ by t, then ψ' is an S-displacement of ψ.

§50

S-displacements of D-sequences

An **S-displacement** of a list L of terms or formulae is obtained by substituting each member of the list L by an S-displacement of it. If the list L is seen as a D-sequence, then its S-displacement L' much fulfill some other conditions, the T-sequence of L' will be a given S-displacement of the T-sequence of L, the G-sequence of L will be an S-displacement of the G-sequence of L.

A D-sequence Δ is completely determined by giving its first formula and, for each step, the selected auxiliary term (T-sequence) or position of the selected subformula of the conjunction (first or second). An **S-displacement** Δ' of the D-sequence Δ is specified by giving an S-displacement of its first formula and an S-displacement of each selected auxiliary term, Δ' is a D-sequence obtained by substituting, from left to right, each formula in Δ by an S-displacement of it: the first formula is substituted by the given S-displacement of it; if the formula φ of Δ was substituted by its S-displacement φ' and ψ follows in Δ, then φ' has the same structure as φ, ψ is substituted by a ψ' obtained from φ' with a similar step as the one with which ψ was obtained from φ, the given S-displacement of the auxiliary term used in Δ is used if φ is universal, the first or second subformula of φ' is selected if φ is a conjunction and the first or respectively the second subformula was selected; the atomic last formula of Δ will be, as the other, substituted by an S-displacement of it, and hence by an atomic formula. Obviously, the T-sequence of Δ' is obtained by substituting each term of the T-sequence of Δ by the given S-displacement of it, and the G-sequence of Δ' by substituting each formula in the G-sequence of Δ by an appropriate S-displacement of it.

§51

S-displacements of mj-rules

An FE-, UE-, OE_i-, AE- or EE-rule is completely determined by giving its undersequent and its auxiliary terms, if it has any. A most precise name for these rules would contain the undersequent and its auxiliary terms. Let r be a rule of one of these schemata, an **S-displacement** of r is specified by giving an S-displacement of its undersequent, not containing the proper variable of r if this is an AE-rule, and an S-displacement of its auxiliary term if it is an EE-rule: it is the only rule of the same schema whose undersequent is the given S-displacement of the one of r, whose proper variable is the same as the one of r if this is an AE-rule, whose auxiliary term is the given S-displacement of the one of r if this is an EE-rule. The oversequents of the S-displacement of r correspond to S-displacements of the oversequents of r.

A similar definition for m-rules is given. An m-rule $m(\Sigma, \Delta)$ is completely determined by giving Σ and Δ. By giving an S-displacement Σ' of the antecedent Σ of the undersequent of $m(\Sigma, \Delta)$ and S-displacements of its auxiliary terms, namely, of the T-sequence of Δ, we can find the S-displacement Δ' of Δ whose T-sequence is the given S-displacement of the T-sequence of Δ and whose first formula is the formula in the given S-displacement Σ' of Σ corresponding to the first formula of Δ (as element of Σ). With the given Σ' and this calculated Δ' we can build the **S-displacement** $m(\Sigma', \Delta')$ of the m-rule $m(\Sigma, \Delta)$. Hence, the S-displacement of an m-rule depends on a given S-displacement of the antecedent of its undersequent and of the given S-displacements of its auxiliary terms; one could give an S-displacement of the whole undersequent, but nothing warranties that it will become the undersequent of the S-displacement of the rule. The oversequents of an S-displacement of an m-rule correspond to S-displacements of the oversequents of the original rule, as is the case with rules of other schemata.

For recursively constructing S-displacements of composition trees, from the root to the leaves, we would like that an S-displacement of a rule be determined by giving an S-displacement of its undersequent and an S-displacement of its auxiliary terms. As seen, this is not the case for m-rules. This forces us to introduce the concepts of *potential and actual mj-rules.*

§52

Potential and actual mj-rules

A **potential rule** is a rule whose oversequents are divided in **proper** and **non-proper** ones, the non-proper ones being of the form $\Sigma \vdash s = t$, where the $s = t$ are equations called **conditions** of the rule. A potential rule all of whose conditions are of the form $s = s$ is called an **actual rule**. In compositions, non-proper oversequents are predestinated to be firstsequents.

A **potential mj-rule** $\bar{r}$ is specified by giving an mj-rule r and, in the case this is an m-rule, additionally an atomic formula whose predicate symbol coincide with the one of the succedent of the undersequent of r. If r is an FE-, UE-, OE-, AE- or AE-rule, then $\bar{r}$ coincide with r, it is called a potential FE-, UE-, OE-, AE- or AE-rule, it does not have non-proper oversequents, it is also an actual FE-, UE-, OE-, AE- or AE-rule. If r is an m-rule $m(\Sigma, \Delta)$ with undersequent $\Sigma \vdash r(t_1, \ldots, t_n)$ and if the given atomic formula is $p(s_1, \ldots, s_n)$, then the non-proper oversequents of $\bar{r}$ are $\Sigma \vdash t_1 = s_1$, $\ldots$, $\Sigma \vdash t_n = s_n$, they are followed by the proper oversequents that coincide with the ones of r, its undersequent is $\Sigma \vdash p(s_1, \ldots, s_n)$. If the undersequent of r contains comp_L in the antecedent, this potential m-rule $\bar{r}$ is the composition of two m-rules: of r with the m-rule whose oversequents are the non-proper oversequents of $\bar{r}$ followed by $\Sigma \vdash p(t_1, \ldots, t_n)$ and whose undersequent is the one of $\bar{r}$, namely, the m-rule $m(\Sigma, \Delta)$, where Δ is the D-sequence for $\text{comp}(p)$ whose T-sequence is $t_1, \ldots, t_n, s_1, \ldots s_n$. The potential m-rule $\bar{r}$ is an actual m-rule if and only if the selected atomic formula is exactly the succedent of the undersequent of r.

Auxiliary terms, proper variable and D-sequence of a potential mj-rule $\bar{r}$ are by definition the ones of the given mj-rule r.

§53

S-displacements of potential mj-rules

Given a potential mj-rule r, an S-displacement $\Sigma' \vdash \xi'$ of its undersequent, not containing the proper variable of r if this is a potential AE-rule, and S-displacements of its auxiliary terms if r is an m- or EE-rule, we define a new potential mj-rule r' called the **S-displacement** of r: it is a potential rule of the same schema as r, its undersequent is the given S-displacement of the undersequent of r, its proper variable is the same as the one of r if it is an AE-rule, its auxiliary term is the given S-displacement of the one of r if it is an EE-rule, its D-sequence Δ' is the S-displacement of the D-sequence Δ of r whose first formula is the formula in Σ' corresponding to the first formula of Δ and whose T-sequence is obtained by substituting each element of the T-sequence of Δ by the given S-displacement of it if r is an m-rule. By considering each schema (FE, UE, OE_i, AE, EE or m) for potential mj-rules, we easily see that there is a unique potential mj-rule r' with the demanded properties.

Furthermore, the S-displacement r' of r is obtained by substituting the undersequent of r by the given S-displacement of it, by substituting each oversequent by an appropriate S-displacement of it, proper oversequents by proper oversequents, non-proper ones by non-proper ones, by substituting each auxiliary term by the given S-displacement of it, where the proper variable in an AE-rule is preserved. Each symbol appearing in an oversequent of r' appears either in its undersequent or in the given S-displacement of an auxiliary term of r'. These properties will allow us to define S-displacements of mj-compositions, and hence of potential mj-derivations.

§54

mj*-compositions, potential mj*-derivations

An **mj*-composition** is a composition with potential mj-rules, in which non-proper oversequents are left free as firstsequents. Definitions and results in §3 and §10 can be trivially extended to mj*-compositions.

There is a trivial one to one correspondence between actual mj-rules and mj-rules, the image of an actual mj-rule $\bar{r}$ is the given r, it is recuperated by deleting the non-proper oversequents. This correspondence can be extended to a one to one correspondence between mj*-compositions built with actual mj-rules and mj-compositions, each actual mj-rule $\bar{r}$ is substituted by its image r, the firstsequents corresponding to non-proper oversequents (their succedents are of the form $s = s$) are deleted.

A potential mj-rule containing comp_L in the antecedent of its undersequent can always be seen as a composition of mj-rules. An mj*-composition whose endsequent contains comp_L in the antecedent is built only with such potential mj-rules; and by substituting each potential mj-rule by the corresponding composition of mj-rules, one gets an mj-composition containing the same endsequents and firstsequents.

A **potential mj*-derivation** is an mj*-composition whose endsequent is a PE-sequent and whose firstsequents have equations as succedents. Between potential mj*-derivations built with actual mj-rules and potential mj-derivations holds the trivial one to one correspondence. Since all PE-sequences contain comp_L in the antecedent, we can substitute each potential mj-rule in a potential mj*-derivation by the corresponding composition of mj-rules for obtaining a potential mj-derivation.

§55

S-displacements of mj*-compositions and potential mj*-derivations, S-equivalence

For an mj*-composition Ξ, an S-displacement of its endsequent not containing proper variables of AE-rules in Ξ, and an S-displacement of each auxiliary term of a potential m- or EE-rule in Ξ, so that it does not contain the proper variable of an AE-rule of Ξ appearing over this m- or EE-rule, we define the **S-displacement** Ξ' of Ξ, as the unique mj*-composition tree obtained by substituting each rule r of Ξ by an *appropriate* S-displacement r' of it, so that the endsequent of Ξ is the given S-displacement of the endsequent of Ξ, so that the S-displacements of the auxiliary terms of r used for building the S-displacement r' of r be the given ones. For building r' we need an S-displacement of its undersequent, this is why Ξ' must be constructed recursively from the root to the leaves: we have an S-displacement of the undersequent of the rule r whose undersequent is the endsequent of Ξ (the given one), we can find r' for this r, its proper oversequents are S-displacements of the oversequents of r, these oversequents are S-displacements of the endsequents of the subtrees of Ξ over them, recursively we can find the corresponding S-displacements of these smaller trees and mount them over the proper oversequents of r'. Although we did not demand that Ξ have unambiguous proper variables, this definition is mainly applied to such derivations.

Since S-displacements of equations are again equations, an S-displacement Ξ' of a potential mj*-derivation Ξ is also a potential mj*-derivation. If S does not contain any proper variable of Ξ or equivalently Ξ' and if R and R' are the most general resolvents of Ξ and Ξ', then $S \cup R$ and $S \cup R'$ are equivalent; we say that R and R' are **S-equivalent**. To see this, let $s' = t'$ be a condition of Ξ' corresponding to a condition $s = t$ of Ξ, and let $\bar{s} = \bar{t}$ and $\bar{s}' = \bar{t}'$ be the equations obtained from $s = t$ and $s' = t'$ by substituting in both the proper variables $q_1, \ldots, q_n$ by some variables $v_1, \ldots, v_n$. These last equations may be seen, up to variable renaming, as elements of R and R'. Since $s' = t'$ is an S-displacement of $s = t$, then $s \rightarrow_2 s'$ and $t \rightarrow_2 t'$ are derivable from S. Since S does not contain the q_i and the open m-schemata used for TRS are stable under the substitutions of q_i by v_i, then also $\bar{s} \rightarrow_2 \bar{s}'$ and $\bar{t} \rightarrow_2 \bar{t}'$ are derivable from S. This implies the equivalence of $S \cup \{\bar{s} = \bar{t}\}$ and $S \cup \{\bar{s}' = \bar{t}'\}$.

3.4 Lifting

§56

Most general potential mj*-derivations

A potential mj*-derivation Ξ is **most general** if (1) it has unambiguous proper variables, (2) auxiliary terms of its potential EE- and m-rules are of form $Xq_1 \cdots q_n$, where X is an unknown and $q_1 \cdots q_n$ is the list of proper variables of the undersequent (and hence of the oversequent, see §10) of the rule, and (3) the operators of different auxiliary terms of EE- or m-rules (of the same rule or of different rules) are formally different unknowns not appearing in the endsequent.

A **specification** of an auxiliary term $Xq_1 \cdots q_n$ of a potential EE- or m-rule r in the most general potential mj*-derivation Ξ is an equation of the form $Xv_1 \cdots v_n = t$, in which t does not contain other variable than the v_i, nor an operator of another auxiliary term, nor a q_i, nor the proper variable of an AE-rule in Ξ. A **specification** S of the auxiliary terms of the most general potential derivation Ξ is a system whose equations are specifications for each auxiliary term. We call the S-displacement Ξ' of Ξ that leaves the endsequent unmoved and S-displaces each auxiliary term $Xq_1 \cdots q_n$ to $t[q_1, \ldots, q_n]$ the **S-evaluation** of Ξ. One can prove by induction that Ξ' is obtained by taking each $Xv_1 \cdots v_n = t[v_1, \ldots, v_n]$ in S and substituting each occurrence of $Xq_1 \cdots q_n$ in Ξ by $t[q_1, \ldots, q_n]$.

§57

Lifting

For a potential mj*-derivation Ξ, we can find, according to §3, a similar potential mj*-derivation Ξ' with the same endsequent and the same (up to variable renamings) most general resolvent. For each auxiliary term t of each EE- or m-rule r in Ξ' let X be a new unknown, neither appearing in Ξ' nor selected for another auxiliary term, let ε be the equation $Xq_1 \cdots q_n = t$, where $q_1 \cdots q_n$ is the list of proper variables of the undersequent of r, and ϵ' be the equation $Xv_1 \cdots v_n = t'$ obtained by substituting the q_i by different variables v_i in ε. Let S be the system whose equations are the ε' corresponding to each auxiliary term in Ξ' and S' the system whose equations are the symmetries $t' = Xv_1 \cdots v_n$ of the equations $Xv_1 \cdots v_n = t'$ in S. We build now an S'-displacement Ξ'' of Ξ': the S'-displacement of the endsequent is the same endsequent of Ξ' and Ξ, the S'-displacement t'' of an auxiliary term t is the corresponding $Xq_1 \cdots q_n$

obtained with $t' = Xv_1 \cdots v_n$. This Ξ'' is a most general potential mj*-derivation and S is a specification of its auxiliary terms, the pair (Ξ'', S) is called a **lifting** of Ξ, from its construction we can easily see that it is unique up to renaming of proper variables, renaming of the unknowns selected for the auxiliary terms and renaming of the variables appearing in the equations of S. The S-evaluation of Ξ'' is exactly the intermediate potential mj*-derivation Ξ' with unambiguous proper variables.

The most general resolvent R of Ξ coincides with the one of Ξ' and is S-equivalent with the most general resolvent R'' of Ξ''. Hence, $S \cup R$ and $S \cup R''$ are equivalent. In particular, for any resolvent T of Ξ holds $S \cup T \vdash S \cup R''$; we say that R'' is **more general** than T **modulo** S. — Since the operators of the left parts of equations in S are different unknowns not appearing in R, since their arguments are variables and since S is cp-closed, the union $S \cup R^*$ of S and a cp-closure R^* of R is cp-closed, it is a cp-closure of $S \cup R$. Hence, $S \cup R$ is consistent if and only R is consistent. And if R is consistent, then also R'' (as subset of $S \cup R''$). Because of this, we search only for most general potential mj*-derivations whose most general resolvent is consistent.

References

[1] Chin-Liang Chang, Richard Char-Tung Lee, "Symbolic Logic and Mechanical Theorem Proving", Academic Press, New York, 1973.

[2] Alonso Church, The calculi of lambda conversion, volume 6 of Annals of Mathematics Studies, Princeton University Press, 1941.

[3] Gottlob Frege, "Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens", Verlag von Louis Nebert, Halle, 1879.

[4] Gerhard Gentzen, "Untersuchungen über das logische Schließen", Mathematische Zeitschrift 39 (1934-5), Seiten 176–210, 405–431.

[5] Ingebrigt Johansson, "Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus", Compositio math. 4, p. 119–136.

[6] Miller, D., Nadathur, G., Pfenning F. and Scedrov, A., Uniform Proofs as a Foundation of Logic Programming, *Annals of Pure and Applied Logic* 51:125–157 (1991).

[7] "Logic Programming in a Fragment of Intuitionistic Linear Logic" J. Hodas and D. Miller, Information and Computation, Volume 110, No. 2, 05.1994.

[8] Stephen Cole Kleene, "Introduction to Metamathematics", Wolters-Noordhoff Publishing - Groningen, North-Holland Publishing Company - Amsterdam, First Published 1952, Sixth Reprint 1971.

[9] Jan Willem Klop, Term Rewriting Systems, In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors: Handbook of Logik in Computer Science, volume 2, chapter 1, pages 1–116. Oxford University Press, 1992.

[10] J. W. Lloyd, "Foundations of Logic Programming", Second, Extended Edition, Springer-Verlag, Berlin, 1987.

[11] A. Martelli und U. Montanari, "An Efficient Unification Algorithm", ACM Transactions on Programming Languages and Systems 4, 2 (April 1982), pages 258-282.

[12] Miller, D., Nadathur, G., Pfenning F. and Scedrov, A., Uniform Proofs as a Foundation of Logic Programming, *Annals of Pure and Applied Logic* 51:125–157 (1991).

[13] G. Mints, "Normal Forms for Sequent Derivations", in Kreiseliana, Editor: P. Odifreddi, Mass. Wellesley, 1996.

[14] M. S. Paterson and M. N. Wegman, "Linear Unification", Journal of Computer and System Sciences 16, 158–167 (1978).

[15] Dag Prawitz, "Natural Deduction", Almqvist & Wiksells, Uppsala, 1965.

[16] Readi Nasser, R. A Reduction Oriented Calculus, Internal Report 293/97, C. S., University of Kaiserslautern, 21.10.1997.

[17] Helmut Schwichtenberg, Minimal from Classical proofs, Note in CSL91.

[18] A. S. Troelstra, H. Schwichtenberg, Basic Proof Theory, Cambridge University Press, 1996.