PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT.

On the Duality of Abduction and Model Generation

Marc Denecker * Danny De Schreye †

Department of Computer Science, K.U.Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium.
e-mail : {marcd, dannyd}@cs.kuleuven.ac.be

Abstract

We present a duality relationship between abduction for definite abductive programs and model generation on the only-if part of these programs. As was pointed out by Console et all, abductive solutions for an abductive program correspond to models of the only-if part. We extend this observation by showing that the procedural semantics of abduction itself can be interpreted dually as a form of model generation on the only-if part. This model generation extends Satchmo with an efficient treatment of equality. It is illustrated how this duality allows to improve current procedures for both abduction and model generation by transferring technical results known for one of these computational paradigms to the other.

1 Introduction

The work we report on this paper was motivated by some recent progress made in the field of Logic Programming to formalize abductive reasoning as logic deduction (see [Console *et al.*, 1991] and [Bry, 1990]). In [Kowalski, 1991], R. Kowalski presents the intuition behind this approach. He considers the following simple definite abductive logic program:

P = { wobbly-wheel ← flat-tyre.
wobbly-wheel ← broken-spokes.
flat-tyre ← punctured-tube.
flat-tyre ← leaky-valve. }

where the predicates broken-spokes, punctured-tube and leaky-valve are the abducibles. Given a query Q = ← *wobbly-wheel*, abductive reasoning allows to infer the assumptions:

S_1 = { punctured-tube },
S_2 = { leaky-valve }, and
S_3 = { broken-spokes } .

These sets of assumptions are abductive solutions to the given query $\leftarrow Q$ in the sense that for each S_i, we have that $P \cup S_i \models Q$.

Kowalski points out that we can equally well obtain these solutions by deduction, if we first transform the abductive program $P \cup \{Q\}$ into a new logic theory T. The transformation consists of taking the only-if part of every definition of a non-abducible predicate in the Clark-completion of P and by adding the negation of Q. In the example, we obtain the (non-Horn) theory T:

T = { wobbly-wheel → flat-tyre, broken-spokes.
flat-tyre → punctured-tube, leaky-valve.
wobbly-wheel ← }

Minimal models for this new theory T are:

M_1 = { wobbly-wheel, flat-tyre, punctured-tube },
M_2 = { wobbly-wheel, flat-tyre, leaky-valve },
and
M_3 = { wobbly-wheel, broken-spokes }.

Restricting these models to the atoms of the abducible predicates only, we precisely obtain the three abductive solutions S_1, S_2 and S_3 of the original problem.

The above observation points to an interesting issue; namely the possibility of linking these dual declarative semantics by completely equivalent dual procedures. Figure 1 shows this duality between an SLD+ Abduction tree (see [Cox and Pietrzykowski, 1986]) and the exectution tree of Satchmo, a theorem prover based on model generation ([Manthey and Bry, 1987]).

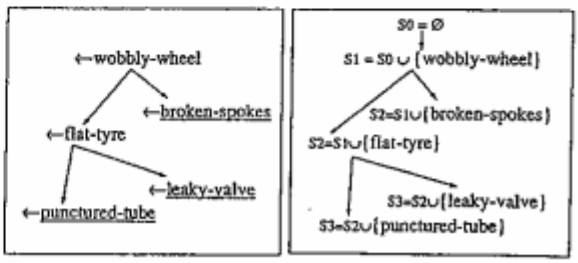

Figure 1: Procedural Duality of Abduction and Satchmo

*supported by the Belgian "Diensten voor Programmatie van Wetenschapsbeleid", under the contract RFO-AI-03
†supported by the Belgian National Fund for Scientific Research

Although this example illustrates the potential of using deduction or more precisely, model generation, as a formalisation of abductive reasoning, an obvious restriction of the example is that it is only propositional. Would this approach also hold for the general case of definite abductive programs? An example of a non-propositional program and its only-if part is given in figure 2.

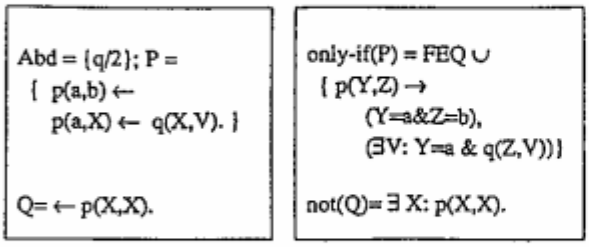

Figure 2: A predicate example

The theory only-if(P) consists not only of the only-if part of the definitions of the predicates but comprises also the axioms of Free Equality (FEQ), also known as Clark Equality ([Clark, 1978]). The abductive solutions and models of only-if(P) are displayed in figure 3.

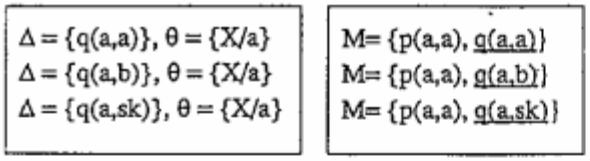

Figure 3: Abductive solutions and models

The duals of the abductive solutions are again identical to models of only-if(P). This example suggests that at least the duality on the level of declarative semantics is maintained. However, on the level of procedural semantics, some difficulties arise. The SLD+Abduction derivation tree is given in figure 4.

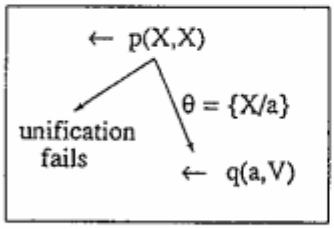

Figure 4: Abductive derivation tree

After skolemisation of the residue $\leftarrow q(a, V)$, we obtain the third abductive solution. With respect to the model generation, the theory only-if(P) is not clausal, however the extension of Satchmo, Satchmo_1 ([Bry, 1990]), can deal with such formulas directly (without normalisation to clausal form). Without dealing with the technical details of the computation, figure 5 presents the computation tree.

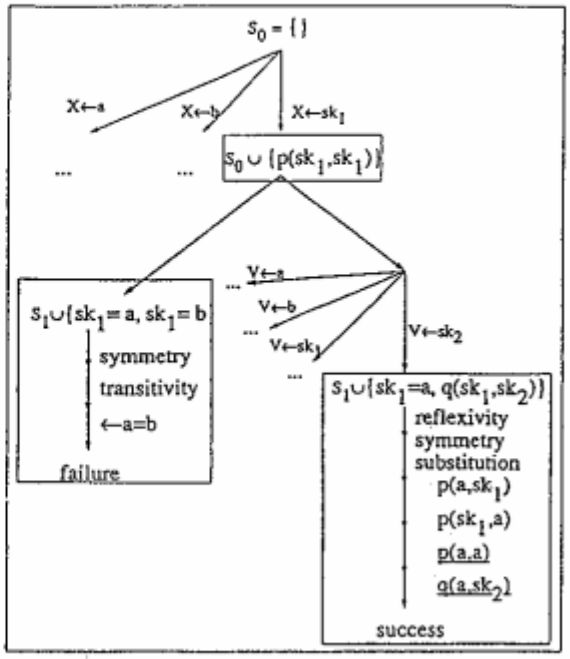

Figure 5: Execution tree of Satchmo_1

Globally, the structure of the SLD+abduction tree of figure 4 can still be seen in the Satchmo_1-tree. Striking is the *duality* of variables in the abductive derivation and skolem constants in the model generation. However, one difference is that the Satchmo_1 tree comprises many additional inference steps due to the application of the axioms of FEQ[1]. In the abductive derivation these additional steps correspond to the unification operation (e.g. on both left-most branches, the failure of the unification of {X=a, X=b} corresponds to the derivation of the inconsistency of the facts $\{sk_1 = a, sk_1 = b\}$).

Another difference is that the generated model

$$\{p(a,a), q(a,sk_2),\ p(sk_1,a), p(a,sk_1), p(sk_1,sk_1),\ q(sk_1,sk_2), sk_1 = a, a = sk_1, a = a, sk_1 = sk_1, sk_2 = sk_2\}$$

is much larger than the model which is dual to the abductive solution. Satchmo_1 generates besides the atoms of this model also all logical implications of FEQ, comprising all substitutions of a by sk_1. It is clear that in general this will lead to an exponential explosion.

However, observe that we obtain the desired model by *contracting* sk_1 and a in the generated model. Therefore, extending Satchmo_1 with methods for dynamic contraction of equal elements would solve the efficiency problem and would restore the duality on the level of declarative semantics.

Contraction of a model is done by taking one unique witness out of every equivalence class of equal terms and

[1]Improper use of Satchmo_1: Equality in head of rule.

replacing all terms in the facts of the model by their witnesses. Techniques from Term Rewriting can be used to implement this. The procedural solution is to consider the set of inferred equality facts as a Term Rewriting System (TRS), to transform the set to an equivalent *complete* TRS, and to *normalise* all facts in the model using this complete TRS, and this after each forward derivation step in Satchmo_1.

This procedure may seem alien to Logic programming, but the contrary is true. As a mather of fact, the proposed procedure appears to be exactly the dual of techniques used in SLD+Abduction:

- the completion procedure corresponds dually to unification.
  The dual of the mgu (by replacing variables by skolem constants) is the completion of the set of equality atoms.
- the normalisation corresponds dually to applying the mgu.

Therefore, incorporating these techniques in Satchmo_1 would also restore the duality on the level of procedural semantics.

The research reported in this paper started as a mathematical exercise in duality. However, there are clearly spinoffs. One application is the extension of Satchmo_1 with efficient treatment of equality. We propose a framework for model generation under an arbitrary equality theory and we formally proof the duality of SLD+abduction in the instance of the framework, obtained by taking FEQ as the equality theory. Also for abduction there are spinoffs. An illustration of this is found in the context of planning as abduction in the event calculus. The event calculus contains a clause, saying that a property holds at a certain moment if there is an earlier event which initiates this property, and the property is not terminated (clipped) in between:

$$holds\_at(P,T) \leftarrow happens(E), initiates(E,P), E < T, \neg clipped(E,P,T).$$

A planner uses this clause to introduce new events which initialise some desired property. Technically this is done by first skolemising and then abducing the *happens* goal. However, skolemisation requires explicit treatment of the equality predicate as an abducible satisfying FEQ ([Eshghi, 1988]). The techniques proposed in this paper allow efficient treatment of the abduced equality atoms, and provide a declarative semantics for it.

The paper is structured as follows. In section 2, we present the class of theories for which the model generation is designed. Section 3 recalls basic concepts of Term Rewriting. In section 4, the framework for model generation is presented and important semantic results are formulated. In section 5, the duality with abductive reasoning is formalised. Section 6 discusses future and related work. Due to space restrictions, all proofs are omitted. We refer to [Denecker and De Schreye, 1991] for the explicit proofs.

2 Extended programs.

In this section we introduce the formalism for which the model generation will be designed. This formalism should at least contain any theory that can be obtained as the only-if part of the definition in the Clark-completion of definite logic programs. The extended clause formalism introduced below, generalises both this kind of formulas and the clausal form.

Definition 2.1 *Let L be a first order language.*
An extended clause *or* rule *is a closed formula of the type:* $\forall(G_1,\ldots,G_k \rightarrow E_1,\ldots,E_l)$
where E_i has the general form:

$$\exists Y_1,..,Y_m : s_1 = t_1 \& \ldots \& s_g = t_g \& F_1 \& \ldots \& F_h$$

such that all G_i are atoms based on L, all F_i are non-equality atoms based on L

Definition 2.2 *An extended program is a set of extended clauses.*

Interestingly, the extended clause formalism can be proven to provide the full expressivity of first order logic. Any first order logic theory can be translated to a logically equivalent extended program, in the sense that they share exactly the same models. (Recall that the equivalence between a theory and its clausal form is much weaker: the theory is consistent iff its clausal form is consistent.)

In the sequel, the theory of general equality (resp. the theory of Free Equality), for a first order language L will be denoted EQ(L) (resp. FEQ(L)). A theory T, based on L, is called a *theory with equality* if it comprises EQ(L). A theory T, based on L is called an *equality theory* if it is a theory with equality in which "=" is the only predicate symbol in all formulas except for the substitution axioms of EQ(L).

3 Concepts of Term Rewriting.

The techniques we intend to develop for dealing with equality, are inspired by Term Rewriting. However, work in this area is too restricted for our purposes, because the concepts and techniques assume the general equality theory EQ underlying the term rewriting. To be able to deal with FEQ, we extend the basic concepts for the case of an arbitrary underlying equality theory E. In the sequel, equality and identity will be denoted distinctly when ambiguity may occur, resp. by "=" and "≡". We assume the reader to be familiar with basic notions of TRS's (see

e.g. [Dershowitz and Jouannaud, 1989]). We just recall some general ideas. A TRS γ associates to each term s a reduction tree in which each branch consists of successive applications of rewrite rules of γ. If γ is *noetherian*, these trees are all finite. If moreover γ is *Church-Rosser* or *confluent*, all leaves of the reduction tree of any term t contain the same term, called the normalisation of t and denoted $t.\gamma$. In Term Rewriting, such a TRS is called *complete*. Below we extend this concept.

Definition 3.1 *Let E be an equality theory based on a language L, γ a Term Rewriting System based on L.*

γ is complete wrt to <L,E> iff γ is noetherian and Church-Rosser and, moreover, <L,E+γ> has a Least Herbrand Model, which consists of all ground atoms $s = t$ constructed from terms in HU(L) such that $s.\gamma \equiv t.\gamma$.

This definition extends the normal definition in Term Rewriting by the third condition. However, for E = EQ, it has been proved that this property is implied by the noetherian and Church-Rosser properties (for a proof see [Huet, 1980]). Of course this is not the case for an arbitrary equality theory (as FEQ).

Definition 3.2 *A completion of a TRS γ wrt <L,E> is:*

- *$\{\Box\}$ if <L,E+γ> is inconsistent*
- *a complete TRS γ_c, such that $<L,E> \models \gamma \leftrightarrow \gamma_c$*

Our framework for model generation is developed for logical theories consisting of two components, an extended program P and an *underlying* equality theory E. This distinction reflects the fact that the model generation mechanism applies only to the extended clauses of P, while E is dealt with in a procedural way, using completion and normalisation. However, in order to make this possible, E should satisfy severe conditions, which are formulated in the following definition.

Definition 3.3 *An* equality theory with completion, *E, based on a language L, is a clausal equality theory equipped with a* language independent *completion procedure.*

The latter condition means that if γ is a ground Term Rewriting System based on an extension L' of L by skolem constants, and γ_c is the completion of γ wrt to <L',E>, then for any further extension L" of L' by skolem constants, γ_c is still the completion of γ wrt <L",E>.

We denote γ_c as TRS-comp(γ).

4 A framework for Model Generation

Informally a model generator constructs a sequence[1] $(Cl_d, j_d)_1^n$, where Cl_d is the ground instance of a rule applied after d steps, and j_d the index indicating the conclusion of Cl_d that was selected, an increasing sequence of sets of *asserted* ground facts $(M_d)_0^n$ of non-equality predicates, a sequence of complete Term Rewriting Systems $(\gamma_d)_0^n$, each of which is equivalent with the set of *asserted* equality facts, and an increasing sequence of sets of skolem constants $(Sk_d)_0^n$, obtained by skolemizing the existentially quantified variables. Formally:

Definition 4.1 *Let L be a language, L_{sk} an infinite countable alphabet of skolem constants, T an extended program based on L consisting of an equality theory with completion E with completion function TRS-comp and P an extended program.*

An Nondeterministic Model Generator with Equality *(NMGE) K is a tuple of four sequences $(Sk_d)_0^n$, $(M_d)_0^n$, $(\gamma_d)_0^n$ and $(Cl_d, j_d)_1^n$ where $n \in \mathbb{N} \cup \{\infty\}$. The sequences satisfy the following conditions:*

1. *$M_0 = Sk_0 = \{\}; \gamma_0 = TRS\text{-}comp(\{\})$*
2. *for each d such that $0 < d \leq n$, Cl_d, j_d, Sk_d, M_d and γ_d are obtained from Sk_{d-1}, M_{d-1} and γ_{d-1} by applying the following steps:*

   (a) Selection of rule and conclusion

   Define LHM_{d-1} as:
   $LHM(<L+Sk_{d-1}, EQ(L)+ M_{d-1}+ \gamma_{d-1}>)$
   Select nondeterministically a ground normal *instance of a rule of P*

   $$Cl_d = "G_1, \ldots, G_k \rightarrow E_1, \ldots, E_l"$$

   such that $G_1, \ldots, G_k$ hold in LHM_{d-1}. If $l = 0$, define $Sk_d = \{\}$, $M_d = \gamma_d = \{\Box\}$ and $n = d$.

   Otherwise, select nondeterministically a conclusion E_j from the head $E_1, \ldots, E_l$. Define $j_d = j$. We say that the rule Cl_d applies [with its j_d'th conclusion].

   (b) Skolemisation

   Let E_{j_d} be of the form: $\exists Y_1, \ldots, Y_m$:
   $s_1 = t_1 \& \ldots \& s_g = t_g \& F_1 \& \ldots \& F_h$
   Replace $Y_1, \ldots, Y_m$ by unique skolem constants $sk_1, \ldots, sk_m$ from $L_{sk} \backslash Sk_{d-1}$. Define $Sk_d = Sk_{d-1} \cup \{sk_1, \ldots, sk_m\}$

   (c) Completion

   Define $\gamma_d = TRS\text{-}comp(\gamma_{d-1} + \{s_1 = t_1, \ldots, s_g = t_g\})$. If γ_d is $\{\Box\}$ then define $M_d = \{\Box\}$ and $n = d$.

[1] $(A_d)_i^n$ denotes a sequence $(A_i, \ldots, A_n)$

(d) Normalisation+Assertion
Define $M_d = M_{d-1}.\gamma_d + \{F_1, \ldots, F_h\}.\gamma_d$, *obtained by computing the normal form of all facts in these sets.*

K is failed if n is finite and $\gamma_n = M_n = \{\Box\}$. *This situation occurs when* Cl_n *is a negative clause, or when* $E+\gamma_{n-1}+\{s_1 = t_1, \ldots, s_g = t_g\}$ *is inconsistent.*
If K is not failed then K is called successful.

Not all NMGE's are interesting. For example, the empty NMGE (({}), ({}), (TRS-comp({})), ()) trivially satisfies the definition of an NMGE, but will not generate a model if P contains one positive extended clause, i.e. an extended clause with empty body. In that case the empty NMGE is an example of an unfair NMGE: there exists a rule with a true body, but which is never applied.

Definition 4.2 *A NMGE K is fair iff K is failed or if the following conditions are satisfied:*

- *K is successful.*
- *If* $Cl = G_1, \ldots, G_k \rightarrow E_1, \ldots, E_l$ *is a ground instance of a rule of P based on* $L+L_{sk}$, *and there exists a d such that Cl is based on* $L + Sk_d$ *and the body of Cl holds in* LHM_d *then there exists a d' such that* $E_1 \vee \ldots \vee E_l$ *holds in* $LHM_{d'}$.

Property 4.1 $(Sk_d)_0^n$ *is a monotonically increasing sequence.* $(LHM_d)_0^n$ *is a monotonically increasing sequence.*

An NMGE performs a fixpoint computation, the result of which can be seen as an interpretation of the language L and, as we later show, a model of <L,P+E>.

Definition 4.3 *The fixpoint of an NMGE K is* $\cup_0^n LHM_d$ *and is denoted by* $K\uparrow$. *The skolem set used by K is* $\cup_0^n Sk_d$ *and is denoted by Sk(K). K*$\uparrow$ *defines an interpretation of L in the following way:*

- *domain: HU(L+Sk(K))*
- *for each constant c of L:* $K\uparrow(c) \equiv c$
- *for each functor f/n of L:* $K\uparrow(f/n)$ *is the function which maps terms* $t_1, \ldots, t_n$ *of* $HU(L + Sk(K))$ *to* $f(t_1, \ldots, t_n)$.
- *for each predicate of L:* $K\uparrow(p/n)$ *is the set of* $p(t_1, \ldots, t_n)$ *facts in* $K\uparrow$.

Corollary 4.1 *If K is a finite successful NMGE K of length n, then* $K\uparrow = LHM_n$

Theorem 4.1 (Soundness) *If K is a fair NMGE, then* $K\uparrow$ *is a model for* $<L,P+E>$ *and P+E is consistent (a fortiori).*

We say that $K\uparrow$ is the model generated by K.

To state the completeness result, we require an additional concept: the NMGE-Tree. Analogously with the concept of SLD-Tree, an NMGE-Tree is a tree of NMGE's obtained by applying all different conclusions of one rule in the descendents of a node.

Definition 4.4 *Let L be a language, E an equality theory with completion, P an extended program based on L, and* L_{sk} *an alphabet of skolem constants.*
An NMGE-Tree (NMGET) T for $<L,P+E>$ *is a tree such that:*

- *Each node is labeled with a tuple (Sk,M,*γ*) where Sk is a skolem set, M a set of non-equality facts based on L+Sk, and* γ *is a ground TRS based on L+Sk.*
- *To each non-leaf N, a ground instance Cl of a rule of P is associated. For each conclusion with index j in the head of Cl, there is an arc leaving from N which is labeled by (Cl,j).*
- *The sequence of labels on the nodes and arcs on each branch of T constitute an NMGE.*

Definition 4.5 *An NMGET is fair if each branch is fair.*

Definition 4.6 *An NMGET is failed if each branch is failed.*

Observe that a failed NMGET contains only a finite number of nodes. Also if T is inconsistent then because of the soundness Theorem 4.1, each fair NMGET is failed.

As a completeness result, we want to state that for any model of P+E, the NMGE contains a branch generating a *smaller* model. In a context of Herbrand models, the smaller-than relation can be expressed by set inclusion. However, because of the existential quantifers and the resulting skolem constants, we cannot restrict to Herbrand models only. In order to define a smaller-than relation for general models, we must have a mechanism to compare models with a different domain. A solution to this problem is provided by the concept of *homomorphism.*

Definition 4.7 *Let* I_1, I_2 *be interpretations of a language L with domains* D_1, D_2.
A homomorphism from I_1 *to* I_2 *is a mapping h:* $D_1 \rightarrow D_2$ *which satisfies the following conditions:*

- *For each functor f/n* $(n \geq 0)$ *of L and* $x, x_1, \ldots, x_n \in D_1$: $x \equiv I_1(f/n)(x_1, \ldots, x_n) \Rightarrow h(x) \equiv I_2(f/n)(h(x_1), \ldots, h(x_n))$
- *For each predicate symbol p/n* $(n \geq 0)$ *of L and* $x_1, \ldots, x_n \in D_1$: $I_1(p/n)(x_1, \ldots, x_n) \Rightarrow I_2(p/n)(h(x_1), \ldots, h(x_n))$

Intuitively a homomorphism is a mapping from one domain to another, such that all positive information in the first model is maintained under the mapping. Therefore the homomorphsims in the class of models of a theory can be used to represent a "...*contains less positive information than...*" relation. We denote the fact that there exists a homomorphism from interpretation I_1 to I_2 by $I_1 \preceq I_2$. This notation captures the intuition that I_1 contains less positive information than I_2.

For NMGET's we can proof the following powerful completeness result.

Theorem 4.2 (Completeness) *Let E be an equality theory with completion, P an extended program, both based on L. Let L_{sk} be an alphabet of skolem constants.*

1. *There exists a fair NMGET for <L,P+E>.*
2. *For each model M of <L,P+E> and each fair NMGET T, there exists a succesful branch K of T such that $K\uparrow \preceq M$.*

We refer to [Denecker and De Schreye, 1991] for a constructive proof of this strong result. As a corollary we obtain the following reformulation of a traditional completeness result.

Corollary 4.2 *If <L,P+E> is consistent then in each fair NMGET there exists a succesful branch.*

If there exists a failed NMGET for <L,P+E>, then <L,P+E> is inconsistent, and all fair NMGET's are failed.

The completeness result does not imply that all models are generated. For example for P = $\{p \leftarrow q\}$, the model {p,q} is not generated by an NMGE. The following example shows that different NMGET's for the same theory might generate different models.

Example $P = \{\ p, q \leftarrow \quad p \leftarrow\ \}$

Depending on which of these clauses is applied first, we get two different nonredundant NMGET's. If $p \leftarrow$ is applied first, then $p, q \leftarrow$ holds already and is not applied anymore. So we get an NMGET with one branch of length 1. On the other hand if $p, q \leftarrow$ was selected first, then two branches exist and we get the solutions {p} and {p,q}.

Therefore it would be interesting if we could characterise a class of models which are generated by each NMGET. The second item of the completeness Theorem 4.2 gives some indication: for any given model M, some succesful branch of the NMGET generates a model with less positive information than M. For the clausal case, models with no redundant positive information are minimal Herbrand Models. From this observation one would expect that for a clausal program, each fair NMGET generates all minimal models. Indeed, the following completeness theorem holds:

Theorem 4.3 (Minimal Herbrand models) *If P is clausal, then for each fair NMGET T, each minimal Herbrand model is generated by a branch in T .*

We have extended the concept of minimal model for general logic theories and proved the completeness of NMGE in the sense that each fair NMGET T generates all minimal models. We refer to [Denecker and De Schreye, 1991].

5 Duality of SLD+Abduction and Model Generation.

The NMGE framework allows to formalise the observations that were made in the introduction. We first introduce the notion of a dualisation more formally.

Definition 5.1 *Let L be a first order language, L_{sk} an alphabet of skolem constants, V_{sk} a dual alphabet of variables such that a bijection $D : L_{sk} \rightarrow V_{sk}$ exists.*

The dualisation mapping D can be extended to a mapping from $HU(L+L_{sk}) \cup HB(L+L_{sk})$ to the set of terms based on $L+V_{sk}$ by induction on the depth of terms:

- *for each constant c of L : $D(c) \equiv c$*
- *for each term $t = f(t_1, \ldots, t_n)$: $D(f(t_1, \ldots, t_n)) \equiv f(D(t_1), \ldots, D(t_n))$*

D can be further extended to any formula or set of formulas. Under dualisation, a ground TRS γ based on $L+L_{sk}$ corresponds to an equation set $D(\gamma)$ with terms based on $L+V_{sk}$. γ is said to be in solved form iff $D(\gamma)$ is an equation set in solved form.

An equation set is in solved form iff it consists of equations $x_i = t_i$, such that the x_i's are distinct variables and do not occur in the right side of any equation. So a TRS is in solved form if the left terms are distinct skolem constants of L_{sk} which do not occur at the right. A TRS in solved form can also be seen as the dual of a variable substitution.

Property 5.1 *Let γ be a TRS in solved form. Then γ is complete wrt to <L,FEQ>.*

Theorem 5.1 (Duality completion - unification) *FEQ(L) is an equality theory with completion. The completion procedure is dual to unification. The dual of the completion of a ground TRS γ based on L+Sk, is the mgu of $D(\gamma)$. Or D(TRS-comp(γ)) = mgu(D(γ)).*

As was observed in the introduction, this duality can be extended further to the complete process of SLD+abduction. On a procedural level, each resolution step corresponds dually to a model generation step. The selection of a goal for resolution corresponds dually to

the selection of the extended rule with its condition instantiated with the dual of the goal. The selection of the clause in the resolution corresponds dually to the selection of the corresponding conclusion in the extended rule. The unification of goal with the head of the clause and the subsequent application of the mgu, corresponds to the completion of the dual equations in the conclusion and the subsequent normalisation.

Now we can formulate the duality theorem for SLD+Abduction ([Cox and Pietrzykowski, 1986]) and Model Generation.

Theorem 5.2 *Let L be a first order language, with an alphabet of variables L_v, L_{sk} an alphabet of skolem constants, and D: $L_{sk} \rightarrow L_v$ a duality bijection between skolem constants and variables. Let P be a definite abductive program based on L.*

For any definite query $\leftarrow Q$, an abductive derivation for $\leftarrow Q$ and P can be dually interpreted as a fair NMGE for only-if(P)+$\exists$(Q). The set of atoms of the generated model, restricted to the abducible predicates is the dual of the abductive solution. The dual of the answer substitution is the restriction of γ_n to the skolem constants dual to the variables in the query.

The following corollary was proved first by Clark ([Clark, 1978]) for normal programs. For the definite case it follows immediately from the theorem above.

Corollary 5.1 *An SLD-refutation for a query $\leftarrow Q$, and a definite program P without abducibles is a consistency proof of $\exists(Q)$+only-if(P). A failed SLD-tree for a ground query $\leftarrow Q$ and P is an inconsistency proof of $\exists(Q)$+only-if(P), and therefore of $\exists(Q)$+comp(P).*

6 Discussion

A current limitation of the duality framework is its restriction to definite abductive programs. In the future we will extend it to the case of normal abductive procedures. The extended framework will then describe a duality between an SLDNF+Abduction procedure and a form of model generation.

The SLDNF+Abduction procedure can be found by proceeding as for the definite case. There we started from pure SLD and definite programs without abduction, we dualised it and obtained the NMGE method, which under dualisation yields an SLD+Abduction procedure. At present we have performed (on an informal basis) the dualisation of SLDNF for normal programs without abduction. Under dualisation, the resulting model generation procedure gives a natural extension of SLDNF for abductive programs. The abductive procedure incorporates skolemisation for non-ground abducibles goals and efficient treatment of abduced equality atoms by the methods presented earlier. Integrity constraints can be represented by adding for any integrity constraint IC, the rule: "$false \leftarrow not(IC)$.",transforming these rules to a normal program using the transformation of Lloyd-Topor ([Lloyd and Topor, 1984]), and adding the literal *not false* to the query.

A prototype of this method has been implemented. An interesting experiment was its extension to an abductive planner based on the event calculus. Our prototype planner was able to solve some hard problems with context dependent events, problems that are not properly solved by existing systems ([Shanahan, 1989], [Missiaen, 1991]).

In [Denecker and De Schreye, 1992], we proved the soundness of the procedure with respect to Completion semantics, in the sense that for any query $\leftarrow Q$ and generated solution Δ:

$$P + \Delta \models Q$$

This implies the soundness of the procedure with respect to the Generalised Stable Model semantics of [Kakas and Mancarella, 1990b]: a generated solution can be extended in a natural way to a generalised stable model of the abductive program. As a completeness result we proved that the procedure generates all *minimal* solutions when the computation tree is finite.

Related to our work, [Bry, 1990] also indicates a relationship between abduction and model generation. However, while we propose a relationship on the object level, there it is argued that abductive solutions can be generated by model generation on the abductive program augmented with a fixed metatheory.

In [Console *et al.*, 1991], another approach is taken for abduction through deduction. An abductive procedure is presented which for a given normal abductive program P and query $\leftarrow Q$, derives an *explanation formula* E equivalent with Q under the completion of P:

$$comp(P) \models (Q \Leftrightarrow E)$$

The explanation formula is built of abducibles predicates and equality only. It characterises all abductive solutions in the sense that for any set Δ of abducible atoms, Δ is an abductive solution iff it satisfies E.

Although this approach departs also from the concept of completion, it is of a totally different nature. In the first place, our approach aims at contributing to the procedural semantics of abduction. This is not the case with the work in [Console *et al.*, 1991]. Another difference is that this approach is restricted to queries with a finite computation tree. If the computation tree contains an infinite branch, then the explanation formula cannot be computed.

In [Kakas and Mancarella, 1990a], an abductive procedure for normal abductive programs has been defined. A restriction of this method is that abducible goals can only be selected when they are ground. As argued in section 1, this poses a serious problem for applications such

as planning. The methods presented here allow to overcome the problem by skolemisation of nonground goals and efficient treatment of abduced equality facts.

Recently, an planning system based on abduction in the event calculus has been proposed in [Missiaen, 1991]. The underlying abductive system incorporates negation as failure, skolemisation for non-ground abducible goals and efficient treatment of abduced equality facts. However, the system shows some problems with respect to soundness and completeness. Experiments indicated that these problems are solved by our prototype planner.

Finally, we want to draw attention to an unexpected application of the duality framework. In current work on abduction, the theory of Free Equality is implicitly or explicitly present. What happens if FEQ is replaced by general equality EQ and the equality predicate is abducible? The result is an uncommon form of abduction illustrated below. Take the program P = $\{r(a)\leftarrow\}$. For this program, the query $\leftarrow r(b)$ has a successful abductive derivation.

$\leftarrow \underline{r(b)}$ $\quad \Delta = \{\}$

$\square$ $\quad \Delta = \{b = a\}$

$\leftarrow r(b)$ succeeds under the abductive hypothesis {b= a}. The duality framework provides the technical support for efficiently implementing this form of abduction. The only difference with normal abduction is that the completion procedure for FEQ -the dual of unification- must be replaced by a completion procedure for EQ, for example Knuth-Bendix completion.

To conclude, we have presented a duality between two computation paradigms. This duality allows to transfer technical results from one paradigm to the other and vice versa. One application that was obtained was an efficient extension of model generation with equality. Transferring these methods back to abduction, we obtained techniques for dealing with non-ground abducible goals and efficient treatment of abduced equality atoms. We discussed experiments indicating that the extension of the duality framework for the case of normal programs is extremely useful for obtaining an abductive procedure for normal abductive programs.

7 Acknowledgements

We thank Krzysztof Apt, Eddy Bevers, Maurice Bruynooghe and Francois Bry for helpful suggestions.

References

[Bry, 1990] F. Bry. Intensional updates: Abduction via deduction. In *proc. of the intern. conf. on Logic Programming 90*, pages 561–575, 1990.

[Clark, 1978] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and databases*, pages 293–322. Plenum Press, 1978.

[Console *et al.*, 1991] L. Console, D. Theseider Dupre, and P. Torasso. On the relationship between abduction and deduction. *journal of Logic and Computation*, 1(5):661–690, 1991.

[Cox and Pietrzykowski, 1986] P.T. Cox and T. Pietrzykowski. Causes for events: their computation and application. In *proc. of the 8th intern. conf. on Automated Deduction*, 1986.

[Denecker and De Schreye, 1991] Marc Denecker and Danny De Schreye. A framework for indeterministic model generation with equality. Technical Report 124, Department of Computer Science, K.U.Leuven, March 1991.

[Denecker and De Schreye, 1992] Marc Denecker and Danny De Schreye. A family of abductive procedures for normal abductive programs, their soundness and completeness. Technical Report 136, Department of Computer Science, K.U.Leuven, 1992.

[Dershowitz and Jouannaud, 1989] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science, vol.B*, chapter 15. North-Holland, 1989.

[Eshghi, 1988] K. Eshghi. Abductive planning with event calculus. In R.A. Kowalski and K.A. Bowen, editors, *proc.of the 5th ICLP*, 1988.

[Huet, 1980] G. Huet. confluent reductions: Abstract properties and applications to term rewriting systems. *journal of the Association for Computing Machinery*, 27(4):797–821, 1980.

[Kakas and Mancarella, 1990a] A.C. Kakas and P. Mancarella. Database updates through abduction. In *proc. of the 16th Very large Database Conference*, pages 650–661, 1990.

[Kakas and Mancarella, 1990b] A.C. Kakas and P. Mancarella. Generalised stable models: a semantics for abduction. In *proc. of ECAI-90*, 1990.

[Kowalski, 1991] R.A. Kowalski. Logic programming in artificial intelligence. In *proceedings of the IJCAI*, 1991.

[Lloyd and Topor, 1984] J.W. Lloyd and R.W. Topor. Making prolog more expressive. *Journal of logic programming*, 1(3):225–240, 1984.

[Manthey and Bry, 1987] R. Manthey and F. Bry. A hyperresolution-based proof procedure and its implementation in prolog. In *proc. of the 11th German workshop on Artificial Intelligence*, pages 221–230. Geseke, 1987.

[Missiaen, 1991] L. Missiaen. *Localized abductive planning with the event calculus*. PhD thesis, Department of Computer Science, K.U.Leuven, 1991.

[Shanahan, 1989] M. Shanahan. Prediction is deduction but explanation is abduction. In *IJCAI89*, page 1055, 1989.