Bottom-up Abduction by Model Generation

Katsumi Inoue* Yoshihiko Ohta* Ryuzo Hasegawa
Institute for New Generation Computer Technology
21F Mita Kokusai Bldg., 1-4-28 Mita, Minato-ku, Tokyo 108, Japan

MakotoNakashima++
Japan Information Processing Development Center
3-5-8 Shibakoen, Minato-ku, Tokyo 105, Japan

Abstract

We investigate two realizations of parallel abductive reasoning systems using the model generation theorem prover MGTP. The first one, called the MGTP+MGTP method, is a co-operative problem-solving architecture in which model generation and consistency checking communicate with each other. There, parallelism is exploited by checking consistencies in parallel. However, since this system consists of two different components, the possibilities for parallelization are limited. In contrast, the other method, called the Skip method, does not separate the inference engine from consistency checking, but realizes both functions in only one MGTP that is used as a generate-and-test mechanism. In this method, multiple models can be kept in distributed memories, thus a great amount of parallelism can be obtained. We also attempt the upside-down meta-interpretation approach for abduction, in which top-down reasoning is simulated by a bottom-up reasoner.

1 Introduction

Abduction, an inference to explanation, has recently been recognized as a very important form of reasoning for various AI problems that deal with commonsense knowledge as well as scientific and engineering knowledge. For example, in diagnosis, plan recognition and design, when we observe the behavior of a system, we want to identify the hypotheses that can explain the observation. Also, in natural language understanding, sophisticated user interfaces and communication among intelligent agents, it is recognized that an explanatory capability may play a crucial role [Charniak and McDermott, 1985]. One of the most popular formalizations of abduction in AI defines an explanation as a set of hypotheses that, if combined with the background theory, logically entails the given observed formula. This deductive-nomological view of abduction has enabled abduction to be implemented using deduction, in particular with resolution-based proof procedures. Along this line, there have been a number of resolution-based abductive systems [Pople, 1973; Cox and Pietrzykowski, 1986; Poole et a/., 1987; Stickel, 1989; Demolombe and Farinas, 1991; Inoue, 1992].

Thus, we can expect that studies on automated abduction may fill the gap between traditional, fast deductive techniques and more advanced, AI-oriented cornmonsense reasoning. From the viewpoint of automated deduction and theorem proving, however, automated abduction is a hard and challenging problem. This is because:

1. Abduction is not a proof-finding problem but a consequence-finding problem (see [inoue, 1992]).
2. Usually, each abductive explanation is required to be consistent with the background theory. While consistency checking is expensive (undecidable in general), it is essential for some practical applications of abduction (design problems, for example), since we are interested in systems that can reject inconsistent theories to obtain acceptable theories.

In this paper, we propose several techniques for implementing abduction that use fast deductive techniques to realize fast abductive systems. In particular:

1. We provide new implementation methods for abduction using model generation theorem provers such as those in [Manthey and Bry, 1988; Fujita and Hasegawa, 199I]. Instead of finding some logical consequences of the given axioms, our methods generate some models of such formulas.
2. These methods are implemented in parallel on a parallel inference machine. Parallelization is an important source for realizing faster abduction.
3. Top-down, goal information is incorporated in these bottom-up procedures. This is an extension of the Magic Set method for deductive databases [Bancilhon et a/., 1986; Bry, 1990] to deal with abduction.

*Address after April 1993: Department of Information and Computer Sciences, Toyohashi University of Technology, 1-1 Hibarigaoka, Tempaku-Cho, Toyohashi 441, Japan.

+Current address: Department of Information Engineering, University of Industrial Technology, 4-1-1 Hashimotodai, Sagamihara, Kanagawa 229, Japan.

++ Address after April 1993: Department of Computer Science and Intelligent Systems, Faculty of Engineering, Oita University, 700 Dannoharu, Oita-shi, Oita 870-11, Japan.

We use the parallel model generation theorem prover MGTP [Fujita and Hasegawa, 1991] that is implemented in the parallel logic programming language KL1 [Ueda and Chikayama, 1990]. Since the MGTP can be used for both testing the (un)satisfiability of an axiom set and generating the minimal models of a range-restricted axiom set, every function necessary for abduction can be realized on it. To this end, we show two different program transformation methods each of which converts an abductive problem into a model generation problem. The basic idea behind such a transformation has also been employed to compute stable models [Gelfond and Lifschitz, 1988] of general and extended (disjunctive) logic programs in [inoue et al, 1992a].

This paper is organized as follows. In Sections 2 and 3, abduction and the MGTP prover are summarized. Section 4 presents two realizations of parallel abductive systems using the MGTP. In Section 5, we evaluate these systems by applying them to a logic circuit design problem. Some extension of the presented abductive systems and related work are discussed in Section 6.

2 Abduction

The definition of abduction we consider here is similar to that proposed in [Poole et al., 1987]. An abductive framework is a pair (Σ, Γ), where Σ is a set of formulas (the background theory) and T is a set of literals (the hypotheses or abducibles). Let G be a closed formula (the goal). A set E of ground instances of T is an explanation of G from (Σ, Γ) if

1. $\Sigma \cup E \models G$, and
2. $\Sigma \cup E$ is consistent.

An explanation of G is minimal if no proper subset E' of E is an explanation of G.

The computation of explanations of G from (Σ, Γ) can be seen as an extension of proof-tin ding by introducing a set of hypotheses from V that, if they could be proved by preserving the consistency of the augmented theories, would complete the proof of G. Alternatively, abduction can be characterized by a consequence-finding problem [Inoue, 1992], in which some literals are allowed to be hypothesized (or skipped) instead of being proved, so that new theorems consisting of only those skipped literals are derived at the end of deductions instead of just deriving the empty clause. In this sense, abduction can be implemented by an extension of a top-down, backward-chaining theorem-proving procedure. For example, Theorist [Poole et al., 1987] and SOL-resolution [Inoue, 1992] are extensions of the Model Elimination theorem proving procedure [Loveland, 1978].

However, there is nothing to prevent us from using a bottom-up procedure to implement abduction. In fact, we have developed an abductive reasoning system called APRICOT/0 [Ohta and Inoue, 1990], which consists of a forward-chaining inference engine and an ATMS [Reiter and de Kleer, 1987]. The ATMS is used to keep track of the results of inference in order to avoid both repeated proofs of subgoals and duplicate proofs on different hypotheses deriving the same subgoals.

Thus, the two reasoning architectures, top-down and bottom-up, are complementary, yet both have merits and demerits for computing abduction. As Inoue [1992] pointed out, SOL-resolution is direct in the sense that it is both sensitive to the given goal clause and restricted to searching only those formulas consisting of candidate hypotheses only. However, top-down reasoning may result in redundant proofs of subgoals. On the other hand, bottom-up reasoning eliminates redundancy, while it may prove subgoals unrelated to the proof of the given goal.

These facts suggest that it is promising to simulate top-down reasoning using a bottom-up reasoner, or to utilize cached results in top-down reasoning. The former simulation has been proposed for definite Horn databases as the Magic Set [Bancilhon et a/., 1986] or upside-down meta-interpretation [Bry, 1990] methods. As Stickel [1991] argues, this approach is better for abduction than the simulation of bottom-up reasoning by a top-down reasoner. This is because caching is more complicated and less effective for abduction since the search space for abduction is larger than that for deduction. Therefore, [Stickel, 1991] attempts the upside-down meta-interpretation approach for abduction for Horn and non-Horn clauses. While Stickel does not consider the consistency of abductive explanations in his procedure, his approach has been extended to abduction for Horn clauses by incorporating consistency checking for a parallel version of APRICOT/0 in [Ohta and Inoue, 1992].

3 MGTP

This section outlines the model generation theorem prover MGTP [Fujita and Hasegawa, 1991; Inoue et al., 1992a] on which our parallel abductive systems are based. The MGTP is a parallel and refined version of SATCHMO [Manthey and Bry, 1988], which is a bottom-up model generation theorem prover that uses hyperresolution and case-splitting on non-unit derived clauses.

Each clause in an axiom set Σ input to the MGTP is expressed in the form:

$$A_1, \ldots, A_m \rightarrow C_{1,1}, \ldots, C_{1,k_1} \mid \ldots \mid C_{n,1}, \ldots, C_{n,k_n}. \quad (1)$$

where A_i's $(1 \leq i \leq m; m \geq 0)$ and $C_{j,l}$'s $(1 \leq j \leq n, 1 \leq l \leq k_j; k_j \geq 1; n \geq 0)$ are atoms, and all variables are assumed to be universally quantified at the front of the clause. Clause (1) represents the formula $(A_1 \wedge \ldots \wedge A_m) \supset (\{C_{1,1} \wedge \ldots \wedge C_{1,k_1}\} \vee \ldots \vee (C_{n,1} \wedge \ldots \wedge C_{n,k_n}))$ in the standard notation of first-order logic. The left-hand side of $\rightarrow$ is called the *antecedent* of the clause, while the right-hand side is called the *consequent* of the clause. When $n = 0$, the clause $A_1, \ldots, A_m \rightarrow$ is called a *negative clause* and means that whenever a ground instance of $A_1 \wedge \ldots \wedge A_m$ is true, there is a contradiction. When $n = 1$ and $k_1 = 1$, the clause $A_1, \ldots, A_m \rightarrow C_{1,1}$ is called a *definite clause*. A *Horn clause* is either a definite clause or a negative clause.

In the following, an *interpretation* is defined as a set of ground atoms as usual, and is often called a *model candidate*. Given a current set $\mathcal{M}$ of model candidates, the MGTP applies the following operations to every model candidate $M \in \mathcal{M}$ and generates the new set $\mathcal{M}'$:

1. **(Model candidate rejection)** If there is a negative clause of the form:

$$A_1, \ldots, A_m \rightarrow$$

and a substitution σ such that $M \models (A_1 \wedge \ldots \wedge A_m)\sigma$, then M is discarded.

2. **(Model candidate extension)** If there is a non-negative clause of the form:

$$A_1, \ldots, A_m \rightarrow C_{1,1}, \ldots, C_{1,k_1} \mid \ldots \mid C_{n,1}, \ldots, C_{n,k_n}$$

and a substitution σ such that $M \models (A_1 \wedge \ldots \wedge A_m)\sigma$ and $M \not\models (C_{i,1} \wedge \ldots \wedge C_{i,k_i})\sigma$ for all $i = 1, \ldots, n$, then $M \cup \{C_{i,1}\sigma, \ldots, C_{i,k_i}\sigma\}$ is in $\mathcal{M}'$ for every $i = 1, \ldots, n$.

3. Otherwise, M is in $\mathcal{M}'$.

Here, we call the process of obtaining a substitution σ a *conjunctive matching* of the antecedent against elements in M. This process does not need full unification if every clause is *range-restricted* [Manthey and Bry, 1988], that is, if every variable in the clause has its occurrence in the antecedent. In this case, since every model candidate constructed by the MGTP contains only ground atoms, it is sufficient to consider matching instead of full unification. The MGTP also improves efficiency by removing redundant conjunctive matching with a *ramified-stack* algorithm [Fujita and Hasegawa, 1991].

In the following, we assume that function symbols in the language are only constants and that the number of constants is finite. Given a set Σ of clauses and the initial set of model candidates $\mathcal{M}_0 = \{\emptyset\}$, the MGTP applies the above two operations to $\mathcal{M}_0$ and generates $\mathcal{M}_1$. This process is repeated as long as $\mathcal{M}_{n+1} \neq \mathcal{M}_n$ holds. If the MGTP cannot apply any operation to some model candidate set $\mathcal{M}_\alpha$ (that is, $\mathcal{M}_\alpha$ is closed under the above two operations), it stops and returns $\mathcal{M}_\alpha$. If $\mathcal{M}_\alpha = \emptyset$, every model candidate has been rejected so that Σ is *unsatisfiable*. Otherwise, since each interpretation $M \in \mathcal{M}_\alpha$ satisfies every ground clause from Σ, M is a *model* of Σ. For example, if Σ consists of the four clauses:

$$\rightarrow R(A), \quad R(x) \rightarrow S(x), \quad R(x) \rightarrow P(x) \mid Q(x),$$
$$Q(x), S(x) \rightarrow ,$$

then using the MGTP we get:
$\mathcal{M}_0 = \{\emptyset\}$, $\mathcal{M}_1 = \{\{R(A)\}\}$, $\mathcal{M}_2 = \{\{R(A), S(A)\}\}$, $\mathcal{M}_3 = \{\{R(A), S(A), P(A)\}, \{R(A), S(A), Q(A)\}\}$, and $\mathcal{M}_4 = \{\{R(A), S(A), P(A)\}\} = \mathcal{M}_\alpha$. Now, let us denote the set of minimal (in the sense of set inclusion of atoms) model candidates from $\mathcal{M}$ as $min(\mathcal{M})$. Then, the set of *minimal models* of Σ can be precisely given by $min(\mathcal{M}_\alpha)$ [Inoue *et al.*, 1992a; Inoue and Sakama, 1993].

4 Abduction by Model Generation

Here, we introduce two realizations of abductive reasoning systems built on the MGTP.[1] Figure 1 illustrates the abstract architecture for these systems. We consider the

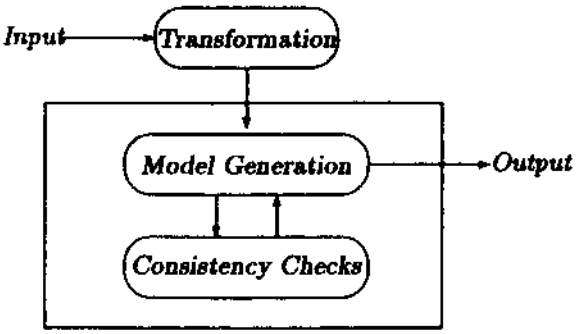

Figure 1: Abduction Architecture based on MGTP

first-order abductive framework (Σ, Γ), where Σ is a set of *range-restricted Horn clauses* and Γ is a set of *atoms* (*abducibles*). Each abductive system first transforms the given abductive framework (Σ, Γ) into a set of clauses in a suitable form. Then, the MGTP is used for both (a) *Model Generation* of the transformed formulas and (b) *Consistency Checks* of some set of formulas augmented with hypotheses. These two functions correspond to the two conditions for each explanation E of G from (Σ, Γ) defined in Section 2: (i) $\Sigma \cup E \models G$ and (ii) $\Sigma \cup E$ is consistent. Here, we convert the problem of finding E by (i) to some model generation problem (a). In this case, since Σ is a set of Horn clauses, each set $\Sigma \cup E$ for any E from Γ is a set of Horn clauses and has a unique minimal model. In this way, the problem of finding explanations of G can be reduced to that of finding models of Σ that satisfy G. The MGTP, thus, outputs every model M of the transformed clauses such that M contains a minimal explanation of the given goal.

The two abductive systems below differ in how they transform the abductive framework (Σ, Γ). The first one, the *MGTP+MGTP method*, is a co-operative system in which *Model Generation* and *Consistency Checks* communicate with each other. The second one, the *Skip method*, on the other hand, does not separate *Model Generation* from *Consistency Checks*, but expands some model candidates while pruning others. Both methods are natural extensions of standard model generation for deduction to model generation for abduction.

4.1 MGTP+MGTP

The MGTP+MGTP method is illustrated in Figure 2. In this method, two kinds of MGTP's are combined: one MGTP is used for *Model Generation*, and the other MGTP's are used for *Consistency Checks*. Then, MGTP-1 works as a forward-inference engine, while MGTP-2 only tests the (un)satisfiability of the given axioms. In this system, each hypothesis H in Γ is represented by $fact(H, \{H\})$, and each *definite* clause in Σ of the form:

$$A_1 \wedge \ldots \wedge A_m \rightarrow C$$

is transformed into a Horn clause of the form:

$$fact(A_1, E_1), \ldots, fact(A_m, E_m) \rightarrow fact(C, cc(\bigcup_{i=1}^{m} E_i)), \quad (2)$$

[1] We have also developed several parallel abductive systems using the MGTP other than the two described in this paper. See [Inoue *et* al., 1992b].

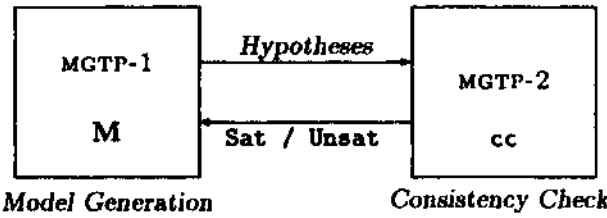

Figure 2: MGTP+MGTP

where E_i is a set of hypotheses from Γ on which A_i depends, and the function cc is defined as:

$$\mathrm{cc}(E) = \begin{cases} E & \text{if } \Sigma \cup E \text{ is consistent;} \\ \mathrm{nil} & \text{otherwise.} \end{cases}$$

Especially, a positive unit clause $\rightarrow C$ is transformed to $\rightarrow fact(C, \emptyset)$. In this case, since each transformed clause is a Horn clause, a global model candidate M is maintained as a unique model candidate in $\mathcal{M}$ at every stage of computation. M contains a set of atomic formulas in the form of $fact(A, E)$ that represents a meta-statement that $\Sigma \cup E \models A$, and is stored in MGTP-1. Each time MGTP-1 derives a new ground atom, the consistency of the combined hypotheses is checked by calling an MGTP-2. [2] Notice that MGTP-1 never uses any negative clause in Σ for *Model Generation*.

In MGTP-2, each clause of Σ is used as is, and, in particular, the negative clauses are necessary. This MGTP is used just for *Consistency Checks*, and whenever a set of hypotheses E is received, it tries to find a model of $\Sigma \cup E$. If a model is found then it returns E itself, else returns nil, which is a code for eliminating the derived *fact*. Note that even when $\Sigma \cup E$ is consistent we do not have to generate all of its minimal models.

The parallelism in this method comes from calling multiple MGTP-2's at once. However, since every transformed clause is Horn, no case-splitting occurs so that MGTP-1 may not be parallelized. Therefore, the effect of parallelization depends heavily upon how much consistency checking can be performed in parallel at once.

4.2 Skip

No matter how good the MGTP+MGTP method might be, the system consists of two different components. The possibilities for parallelization therefore remain limited. In contrast, the Skip method does not separate the inference engine from consistency checking, but realizes both functions in only one MGTP. In this method, the MGTP is used not only as an inference engine but also as a "generate-and-test" mechanism so that consistency checking is automatically performed. We can utilize the capability supplied by the MGTP to extend and reject model candidates. Therefore, multiple model candidates can be kept in distributed memories instead of keeping a single, global model candidate M as in the MGTP+MGTP method. Thus, a great amount of OR-parallelism induced by case-splitting can be obtained.

2Note that transformation (2) is similar to the method proposed by Stickel [1991], except that we additionally consider the function cc for consistency checking.

The most direct way to implement reasoning with hypotheses is as follows. For each hypothesis H in Γ, we supply a clause of the form:

$$\rightarrow H \mid \neg \mathsf{K} H\,, \tag{3}$$

where $\neg \mathsf{K} H$ means that "H is not assumed to be true in the model". Namely, each hypothesis is assumed either to hold or not to hold. Note here that we consider each literal $\neg \mathsf{K} H$ not as a new formula in a suitable modal logic, but as a newly introduced *atom* in a program, hence we are still within the classical first-order logic. Then, for dealing with $\neg \mathsf{K} H$, we need the axiom schema:

$$\neg \mathsf{K} H,\ H \rightarrow \qquad \text{for every hypothesis } H\,. \tag{4}$$

The above schema is an integrity constraint saying that H cannot be true and disbelieved at the same time. In this way, each model candidate containing both H and $\neg \mathsf{K} H$ is rejected.

The above technique, however, may generate $2^{|\Gamma|}$ model candidates, and is, therefore, often explosive for a number of practical applications. To reduce the number of generated model candidates as much as possible, we can use a method that delays case-splitting for each hypothesis. [3] That is, we do not supply any clause of the form (3) for any hypothesis of Γ, but, instead, introduce hypotheses when they are necessary. When abducibles $H_1, \ldots, H_n$ $(n \geq 0)$ from Γ appear in the antecedent of a Horn clause in Σ as:

$$A_1 \wedge \ldots \wedge A_l \wedge \underbrace{H_1 \wedge \ldots \wedge H_n}_{\textit{abducibles}} \rightarrow C\,,$$

we transform this clause into a non-Horn clause:

$$A_1, \ldots, A_l \rightarrow H_1, \ldots, H_n, C \mid \neg \mathsf{K} H_1 \mid \ldots \mid \neg \mathsf{K} H_n\,. \tag{5}$$

In this transformation, each hypothesis H_j in the antecedent is shifted to the right-hand side of the clause in the form of $\neg \mathsf{K} H_j$. Moreover, each H_j is *skipped* instead of being resolved, and is added to consequent C of the rule since C becomes true whenever all A_i's and H_j's are true. This operation is a bottom-up counterpart of the **Skip & Cut** rule in the top-down approach defined for SOL-S resolution in [Inoue, 1992, Section 5.2]. Just like the transformation in (3), we need to supply the schema (4) for the Skip method. For example, suppose that Σ consists of the clauses:

$$\begin{array}{l} \mathit{Person}(x),\ \mathit{Cold}(x) \rightarrow \mathit{Sneeze}(x), \\ \mathit{Person}(x),\ \mathit{Hayfever}(x) \rightarrow \mathit{Sneeze}(x), \\ \rightarrow \mathit{Person}(\mathit{Tom}), \\ \mathit{Person}(x),\ \mathit{Cold}(x),\ \mathit{Hayfever}(x) \rightarrow , \end{array}$$

and the abducibles are $\Gamma = \{\, \mathit{Cold}(x),\ \mathit{Hayfever}(x) \,\}$. Then, (Σ, Γ) is transformed to the following clauses Δ by the Skip method:

$$\begin{array}{l} \mathit{Person}(x) \rightarrow \mathit{Cold}(x),\ \mathit{Sneeze}(x) \mid \neg \mathsf{K}\, \mathit{Cold}(x), \\ \mathit{Person}(x) \rightarrow \mathit{Hayfever}(x),\ \mathit{Sneeze}(x) \mid \neg \mathsf{K}\, \mathit{Hayfever}(x), \\ \rightarrow \mathit{Person}(\mathit{Tom}), \\ \mathit{Person}(x) \rightarrow \neg \mathsf{K}\, \mathit{Cold}(x) \mid \neg \mathsf{K}\, \mathit{Hayfever}(x) \\ \neg \mathsf{K} H,\ H \rightarrow \qquad \text{for every hypothesis } H\,. \end{array}$$

[3] The basic ideas behind the schema (4) and the delayed approach was first proposed for the processing of *negation as failure* with the MGTP in [Inoue *et al.*, 1992a].

Let G = Sneeze(Tom) be the observation. Using the MGTP, two minimal models containing G are obtained from Δ as:

$$M_1 = \{Person(Tom), Cold(Tom), Sneeze(Tom), \neg K\, Hayfever(Tom)\};$$
$$M_2 = \{Person(Tom), Hayfever(Tom), Sneeze(Tom), \neg K\, Cold(Tom)\}.$$

By extracting the abducibles from M\ and M2, we can get the two explanations of G, $E_1 = \{Cold(Tom)\}$ and $E_2 = \{Hayfever(Tom)\}$.

4.3 Upside-Down Meta Interpretation for Abduction

As discussed in Section 2, bottom-up abductive systems can enhance their efficiency by incorporating the goal information and by simulating top-down reasoning. Here, we present two program transformation methods based on upside-down meta-interpretation (UDM) approaches defined by [Bry, 1990; Stickel, 1991]. We first apply such a UDM transformation to the input clauses Σ, generating Σ', then further transform (Σ', Γ) by using a transformation for the MGTP+MGTP method or the Skip method defined in previous subsections.

The first transformation, called the simple UDM transformation, transforms each definite clause of Σ in the form:

$$A_1, \ldots, A_m \rightarrow C \qquad (6)$$

into the clauses:

$$goal(C), A_1, \ldots, A_m \rightarrow C,$$
$$goal(C) \rightarrow goal(A_1), \ldots, goal(A_m).$$

In this method, the MGTP operations are applied for clause (6) only when goal(C) is present. Further, goal(C) invokes the subgoals of (6) by deriving every goal(A_i). Thus, top-down reasoning is simulated in a breadth-first manner.

For each negative clause $A_1, \ldots, A_m \rightarrow$ in Σ, we can apply the above transformation used for definite clauses. Then, since the consequent of a negative clause is empty, we supply $\rightarrow$ goal(A_i) for every A_i. This means that any subgoal in every negative clause is evaluated. However, as Ohta and Inoue [1992] pointed out, many negative clauses are irrelevant to finding explanations of the given goal from the abductive framework in general. If we evaluate all the subgoals in all negative clauses, the UDM method cannot achieve speedups compared with non-controlled bottom-up abduction. To overcome this difficulty, we restrict the evaluation of negative clauses to those clauses relevant to the goal by using the abstracted dependency analyzer [Ohta and Inoue, 1992] that analyzes logical dependencies between the goal and negative clauses at the abstract (e.g., predicate symbol) level.

The second transformation, called the left-to-right UDM transformation, simulates top-down reasoning in such a way that for clause (6) each subgoal goal(A_i+i) $(1 \leq i \leq m-1)$ is invoked only after the previous subgoals goal(A\),...,goal(A_i) have been solved. Thus, this method simulates ordered-linear resolution in a depth-first manner. In this method, each clause of form (6) is transformed to:

$$goal(C) \rightarrow goal(A_1), cont_{k,1}(V),$$
$$cont_{k,1}(V), A_1 \rightarrow goal(A_2), cont_{k,2}(V),$$
$$\vdots$$
$$cont_{k,m}(V), A_m \rightarrow C,$$

where k is the identifier of each clause C in Σ, and $cont_{k,i}(V)$ keeps the obtained substitutions V for the variables appearing in $A_1, \ldots, A_m$. As in the simple UDM transformation, the abstracted-dependency analysis is also employed for evaluating negative clauses in this transformation.

Table 1: Evaluation of Abductive Systems

Table 1-1: Normal Transformation

Problem	limit	MGTP+MGTP	Skip
GCD	300-40	4.0/ 7.5(64)	7.2/ 13.7(64)
	400-50	31.2/ 3.3(64)	23.3/ 19.4(64)
	500-60	54.3/ 2.6(64)	26.1/ 20.6(64)

Table 1-2: Simple UDM Transformation

Problem	limit	MGTP+MGTP	Skip
Subtractor	500-60	1.1/ 1.3(64)	0.6/ 1.2(64)

Table 1-3: Left-to-Right UDM Transformation

Problem	limit	MGTP+MGTP	Skip
Subtractor	500-60	0.8/ 1.0(16)	12.7/ 3.1(64)

[Notation] T / $R(PEs)$
T : The smallest execution time (sec)
R : The speedup ratio of T to the time obtained with a single PE (times)
PEs : The number of PEs used when T is obtained (number)

5 Evaluation

This section presents an evaluation of two abductive systems, the MGTP+MGTP method and the Skip method, by applying them to a design problem. The problem is to design a logic circuit that calculates the greatest common divisor (GCD) of two integers expressed in eight bits by using the Euclidean algorithm [Maruyama et a/., 1988]. The solutions are those circuits satisfying the given constraints on the basic cell count and delay time (the area-time limit). There are several kinds of knowledge on the design of circuits (about 120 clauses): datapath design knowledge (i.e., how to construct a GCD circuit by combining components) at the top level, component design knowledge (e.g., a subtractor can be constructed from the combination of a one's complement circuit and an adder) at the next level, and technology mapping rules (e.g., an adder can be constructed from a series of some CMOS standard cells) at the low level. The problem can be represented as abduction, in which we assume that some combination of components may satisfy all constraints. Thus, if hypotheses derive a contradiction with

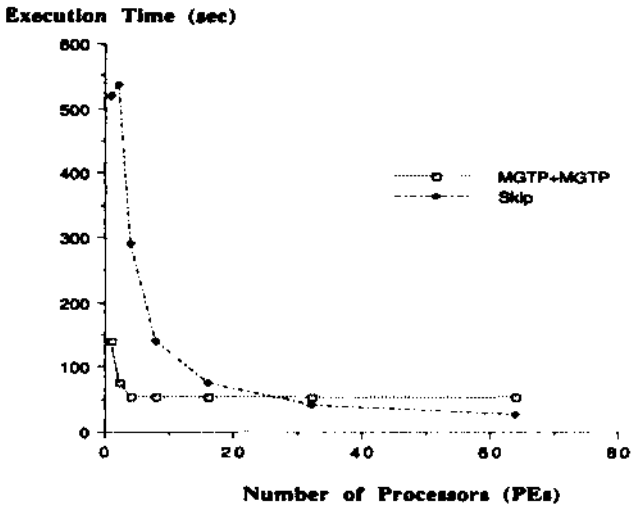

Figure 3: Execution Time *(Circuit: limit 500-60)*

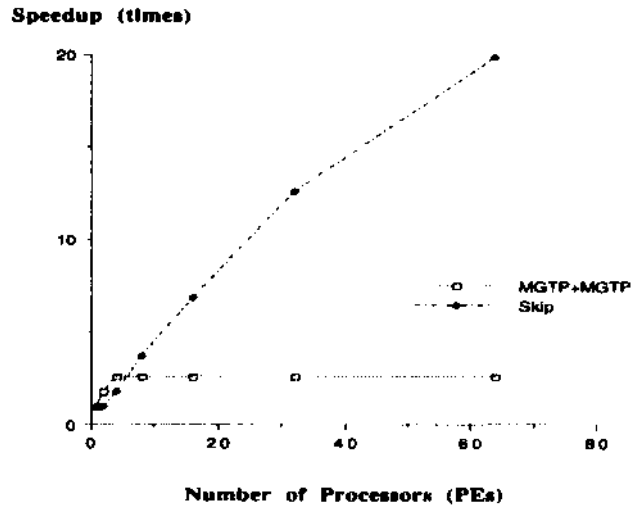

Figure 4: Speedup Ratio *(Circuit: limit 500-60)*

the background knowledge, we see that the sub-design violates some constraints.

Table 1 shows the experimental result of two abductive systems at run time on the PIM/m parallel inference machine developed at ICOT. The area-time limits are set in three ways (300-40ns, 400-50ns and 500-60ns). In order to evaluate two UDM methods, we also solved a subproblem (the design of subtractors) of the entire problem (the design of GCD circuits). [4]

As shown in the table, the run time for designing subtractors in each system is, as expected, much shorter than that for designing GCD circuits, since the reasoning is directed to the given subgoal. However, for the Skip method, the simple UMD transformation works better than the left-to-right UDM transformation, indicating that the left-to-right simulation of top-down reasoning increases the sequential processing more than the breadth-first manner.

Figure 3 shows the run time graph for the design of GCD circuits with a 500-60ns limit, which is obtained by varying the number of available processor elements (PEs) between 1 and 64 on PIM/m. Figure 4 displays the speedup ratio for the same problem when running two abductive systems. All reasoning tasks split with disjunctions are automatically allocated to the available number of processors. As shown in these figures, the Skip method provides better parallelism as well as faster abductive reasoning than the MGTP+MGTP method.

6 Final Remarks

We have presented several new program transformation techniques for fast, parallel and bottom-up abduction. First, we have converted the Horn abductive problem with consistency checking into model generation problems in two ways. Second, we have applied the two kinds of upside-down meta-interpretation transformations to abduction to incorporate top-down information. Although we need to further investigate how to avoid possible combinatorial explosion in constructing model candidates for the Skip method, we conjecture that the Skip method will be the most promising from the viewpoint of Oil-parallelism.

For related work on computing abduction using a model generation theorem prover, Denecker and De Schreye [1992] recently proposed a proof procedure for *object-level abduction* defined in [Console *et* a/., 1991]. Their abduction does not consider the consistency of explanations. But, in contrast to us, they compute the models of the only-if part of a completed program that is not range-restricted in general even if the original definite clauses are range-restricted. To this end, they have to extend the model generation method by incorporating complex term rewriting techniques, while we can use the original MGTP without change in the Skip method.

While we restricted the abductive framework (£, *T)* to a pair of Horn clauses and atomic abducibles, we can consider an extension of abduction which captures nonmonotonic and default reasoning. Then, another important advantage of the Skip method is that it may easily be combined with *negation as failure* so that knowledge bases can contain both abducibles and negation-as-failure formulas as in the framework of [Kakas and Mancarella, 1990]. This extension of the abductive framework by incorporating negation as failure is formally discussed in [Inoue and Sakama, 1993].

Finally, efforts should be also devoted to investigating extensions of bottom-up first-order abduction to deal with non range-restricted, non-Horn clauses and literal abducibles, which can be dealt with by a top-down approach like SOL-resolution [inoue, 1992]. An example of bottom-up abduction for non-Horn clauses without consistency checking can be found in [Stickel, 199l], which uses contrapositives in the form of definite clauses. This technique may be incorporated for the MGTP+MGTP

[4] In order to avoid possible combinatorial explosion caused by generating many redundant model candidates containing non-minimal explanations, we introduced some negative clauses that can be generated automatically by analysis in the Skip method. See [Inoue *et* o/., 1992b] for details.

method with consistency checking. One difficulty lies in the fact that, since the definite transformation using the meta predicate fact does not involve any case-splitting as is the case in the MGTP+MGTP method, it needs AND-parallelisin [Hasegawa et al., 1992] for speed-up.

Acknowledgment

We wish to thank Mark Stickel and Francois Bry for useful comments on earlier work. We would also like to thank all the members of ICOT PAB (Parallel Abduction) Working Group for discussions on this work.

References

[Bancilhon et al., 1986] F. Bancilhon, D. Maier, Y. Sagiv and J.D. Ullman. Magic sets and other strange ways to implement logic programs. In: Proc. 5th ACM SIGMOD-SIGACT Symp. Principles of Database Systems, pages 1 15, 1986.

[Bry, 1990] Francois Bry. Query evaluation in recursive databases: bottom-up and top-down reconciled. Data & Knowledge Engineering, 5:289-312, 1990.

[Charniak and McDermott, 1985] Eugene Chamiak and Drew McDermott. Introduction to Artificial Intelligence. Addison-Wesley, 1985.

[Console et al., 1991] Luca Console, Daniele Theseider Dupre and Pietro Torasso. On the relationship between abduction and deduction. J. Logic and Computation, l(5):661-690, 1991.

[Cox and Pietrzykowski, 1986] P.T. Cox and T. Pietrzykowski. Causes for events: their computation and applications. In: Proc. 8th Int. Conf. Automated Deduction, Lecture Notes in Computer Science, 230, pages 608 621, Springer, 1986.

[Demolombe and Farinas, 1991] Robert Demolombe and Luis Farinas del Cerro. An inference rule for hypothesis generation. In: Proc. IJCAI-91, pages 152 157, 1991.

[Denecker and De Schreye, 1992] Marc Denecker and Danny De Schreye. On the duality of abduction and model generation. In: Proc. Int. Conf. Fifth Generation Computer Systems 1992, pages 650-657, 1992.

[Fujita and Hasegawa, 1991] Hiroshi Fujita and Ryuzo Hasegawa. A model generation theorem prover in KL1 using a ramified-stack algorithm. In: Proc. 8th Int. Conf. Logic Programming, pages 494-500, 1991.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In: Proc. 5th Int. Conf. Symp. Logic Programming, pages 10701080, 1988.

[Hasegawa et al., 1992] R. Hasegawa, M. Koshimura and H. Fujita. MGTP: a parallel theorem prover based on lazy model generation. In: Proc. 11th Int. Conf. Automated Deduction, Lecture Notes in Artificial Intelligence, 607, pages 776-780, Springer, 1992.

[inoue, 1992] Katsumi Inoue. Linear resolution for consequence finding. Artificial Intelligence, 56(2-3):301-353, 1992.

[Inoue and Sakama, 1993] Katsumi Inoue and Chiaki Sakama. Transforming abductive logic programs to disjunctive programs. In: Proc. 10th Int. Conf. Logic Programming, 1993 (to appear).

[Inoue et al, 1992a] Katsumi Inoue, Miyuki Koshimura and Ryuzo Hasegawa. Embedding negation as failure into a model generation theorem prover. In: Proc. 11th Int. Conf. Automated Deduction, Lecture Notes in Artificial Intelligence, 607, pages 400-415, Springer, 1992.

[inoue et a/., 1992b] Katsumi Inoue, Yoshihiko Ohta, Ryuzo Hasegawa and Makoto Nakashima. Hypothetical reasoning systems on the MGTP. Technical Report TR-763, ICOT, August 1992 (in Japanese).

[Kakas and Mancarella, 1990] A.C. Kakas and P. Mancarella. Generalized stable models: a semantics for abduction. In: Proc. ECAI-90, pages 385-391, 1990.

[Loveland, 1978] Donald W. Loveland. Automated Theorem Proving: A Logical Basis. North-Holland, 1978.

(Manthey and Bry, 1988] Rainer Manthey and Frangois Bry. SATCHMO: a theorem prover implemented in Prolog. In: Proc. 9th Int. Conf. Automated Deduction, Lecture Notes in Computer Science, 310, pages 415 434, Springer, 1988.

[Maruyama et al., 1988] Fumihiro Maruyama et al CoLODEX: a cooperative expert system for logic design. In: Proc. Int. Conf. Fifth Generation Computer Systems 1988, pages 1299 1306, 1988.

[Ohta and Inoue, 1990] Yoshihiko Ohta and Katsumi Inoue. A forward-chaining multiple context reasoner and its application to logic design. In: Proc. 2nd IEEE Int. Conf Tools for Artificial Intelligence, pages 386 392, 1990.

[Ohta and Inoue, 1992] Yoshihiko Ohta and Katsumi Inoue. A forward-chaining hypothetical reasoner based on upside-down meta-interpretation. In: Proc. Int. Conf. Fifth Generation Computer Systems 1992, pages 522 529, 1992. An extended version is to appear in New Generation Computing, 11(3-4), 1993.

[Poole et a/., 1987] David Poole, Randy Goebel and Romas Aleliunas. Theorist: a logical reasoning system for defaults and diagnosis. In: N. Cercone and G. McCalla, editors, The Knowledge Frontier: Essays in the Representation of Knowledge, pages 331 352, Springer, 1987.

[Pople, 1973] Harry E. Pople, Jr. On the mechanization of abductive logic. In: Proc. IJCAI-73, pages 147 152, 1973.

[Reiter and de Kleer, 1987] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: preliminary report. In: Proc. AAAI-87, pages 183 187, 1987.

[Stickel, 1989] Mark E. Stickel. Rationale and methods for abductive reasoning in natural-language interpretation. In: Proc. Int. Scientific Symp. Natural Language and Logic, Lecture Notes in Artificial Intelligence, 459, pages 233 252, Springer, 1990.

[Stickel, 1991] Mark E. Stickel. Upside-down meta-interpretation of the model elimination theorem-proving procedure for deduction and abduction. Technical Report TR-664, ICOT, July 1991.

[Ueda and Chikayama, 1990] Kazunori Ueda and Takashi Chikayama. Design of the kernel language for the parallel inference machine. Computer J., 33(6):494-500, 1990.