O P T I M A L R U N T I M E O P T I M I Z A T I O N

PROVED BY A NEW LOOK AT ABSTRACT INTERPRETATIONS

Bernhard Steffen
Institut für Informatik
Universität Kiel, D-2300 Kiel

Abstract

A two stage *run time optimization algorithm* is presented that combines two well-known techniques in a *Herbrand optimal* manner:
- Kildall's iterative method for *data flow analysis* and
- Morel/Renvoise's *partial redundancy elimination* algorithm.

To combine these techniques in such an optimal way, we firstly have to elaborate Kildall's approach. This is done by means of a new classification method for *abstract interpretations* which has to be introduced before

Secondly we have to extend Morel/Renvoise's technique, which is only conceived to treat the occurrences of a single term, to work on the *value equivalence classes* delivered by the Kildall-like data flow analysis algorithm mentioned above.

Our algorithm being optimal with respect to the *Herbrand interpretation*, it is a well-founded basis for the construction of further algorithms using special properties of a given interpretation. These can be obtained by transforming the Kildall-like analysis stage only.

I. Preface

High-level languages support a convenient and reliable programming, but the required compilers often produce inefficient codes. For example the procedure mechanism and the macro expansion mechanism lead to run time computations being too complicated or even redundant. To avoid this, modern compilers are constructed using *optimizing techniques*. Here the following methods are very important:

- loop invariant code motion and
- common subexpression elimination.

Usually *optimizers* operate on *nondeterministic flow graphs* delivered by the compiler front ends. Based on this representation it is possible to combine and to improve the *optimization techniques* mentioned above in a systematic manner, to receive an algorithm which transforms programs into a minimal form w.r.t. the *Herbrand interpretation* [Gr]. This algorithm mainly consists of a two-stage iterative analysis process:

Firstly, we have a *data flow analysis algorithm* which is based on Kildall's iterative analysis technique [Ki1 and Ki2]. It partitions the occurrences of the program terms in a *Herbrand optimal* manner. This optimality is proved by a new *classification approach* for *abstract interpretations*.

Secondly, we use an algorithm which determines the optimal *locations* for the computations of the source program w.r.t. the equivalence relation delivered by the first analysis stage. This algorithm is a generalization of the *partial redundancy elimination process* stated by Morel/Renvoise in 1979 [MR].

Our algorithm being optimal w.r.t. the *Herbrand interpretation*, it is a well-founded basis for the construction of further algorithms using special properties of a given interpretation. These can be obtained by transforming the first analysis stage only.

We now visualize basis and goal of our algorithm:

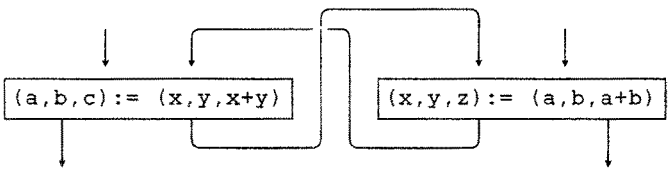

Diagram I.1

Here: - The arrows denote the nondeterministic branching structure of this program part.

- The nodes characterize parallel assignments.

This *non reducible* program part (it is a minimal example to demonstrate the generality of our concept) has the following specific property:

while staying within this general loop (without leaving in the meantime) the computations of "a+b" and "x+y" deliver the same values.

This motivates the following optimization:

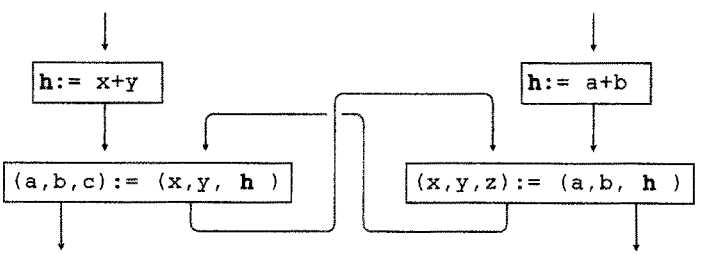

Diagram I.2

This optimization is realized in three steps:

- Recognizing the "equality" of "a+b" and "x+y". This is done within the first analysis stage.
- Determining the *computation points* (see the inserted nodes for the initialization of **h**) and the *computation forms* (here "x+y" and "a+b"). This is realized by solving a *Boolean equation system* initialized by means of the first analysis stage.
- Transforming the program on basis of the analysis results, i.e. moving the computations (here "a+b" and "x+y") to the computation points.

Previous optimization techniques fail in treating the situation mentioned above. Indeed methods exist which are able to determine the redundancy if the variable **h** is already initialized [Ki1, Ki2, KU, FKU and RL], but these methods are either too weak to move any computation at all [Ki1, Ki2 and KU], or they are restricted to special loops [FKU and RL].

On the other hand, a technique exists which is conceived to move a given term within an arbitrary flow graph [MR], but this technique fails to recognize the "equality" of "a+b" and "x+y".

Our approach combines and generalizes Kildall's technique for the redundancy elimination [Ki1 and Ki2] and the method to locate computations within a flow graph stated by Morel/Renvoise [MR]. The result is a uniform algorithm which delivers the optimal *computation points* and *computation forms* w.r.t. the *Herbrand interpretation*.

After presenting the basic concepts of this paper in **section II**, we analyze the relations between different *abstract interpretations* [CC1] in **section III**. The study of these relations delivers *classification criteria* for *abstract interpretations* w.r.t. a given *observational level*. These criteria can be used to prove the *Herbrand optimality* of the *first analysis stage* of our optimization algorithm as sketched in **section IV**. Afterwards, we shortly describe the *second analysis stage* in **section V** and the resulting *optimization* in **section VI**.

II. Preliminary Definitions

We directly introduce the syntax of our programming language as a flow graph structure over a set **N**, which represents the entire set of occurrences of elementary statements. This structure allows flow graphs $G = (N, E, \mathbf{s}, \mathbf{e})$ which are composed as follows:

- (N,E) is a connected and directed graph with node set $N \subseteq \mathbf{N}$ and edge set $E \subseteq N \times N$.
- **s** resp. **e** (*start* resp. *end* node) denotes a special element of N possessing no predecessor resp. successor nodes.
- Further, we write **P**(G) for the set of all finite paths from **s** to **e** of a given flow graph G, **FG** for the set of all flow graphs over **N** and **LFG** for the set of all *linear* flow graphs over **N**, i.e.

  $\mathbf{LFG} = \{ G \in \mathbf{FG} \mid |\mathbf{P}(G)| = 1 \}$.

Remark II.1
P(G) characterizes a special subset of **LFG**. In the following, we identify **P**(G) with this subset.

The *semantics* for flow graphs G over **N**, will now be introduced as the *MOP-solution* in the sense of Kam/Ullman [KU]. For that purpose we need a *complete semi-lattice* $(C, \sqcap, \sqsubseteq, \bot, \top)$ with bottom element $\bot$

and top element $\top$. The elements of this semi-lattice serve as representation of the situations which will possibly occur at an arbitrary program point of G.

Definition II.1

a) Let $[\![\ ]\!]_I : \mathbf{N} \longrightarrow (C \longrightarrow C)$ be a function, which relates to every node $n \in \mathbf{N}$ a distributive function $[\![n]\!]_I$ from C to C (i.e.: $\forall\, T \subseteq C$: $[\![n]\!]_I(\sqcap T) = \sqcap \{\, [\![n]\!]_I(c) \mid c \in T \,\}$). In this case, we call $([\![\ ]\!]_I, C)$ an *abstract interpretation* or a *local abstract semantics*.

b) When $([\![\ ]\!]_I, C)$ is a local abstract semantics and $[\![\ ]\!]$ is the following extension of $[\![\ ]\!]_I$ to the domain **FG**:

$$\forall\, G \in FG,\ c \in C:$$

$$[\![\, G \,]\!](c) =_{df} \begin{cases} [\![n_k]\!]_I \circ \ldots \circ [\![n_1]\!]_I(c), & \text{if } G = (n_1, \ldots, n_k) \in \mathbf{LFG} \\ \sqcap_{P \in \mathbf{P}(G)} [\![P]\!]_I(c) & \text{otherwise,} \end{cases}$$

then $([\![\ ]\!], C)$ is called a (*global*) *abstract semantics* (w.r.t. $([\![\ ]\!]_I, C)$).

This approach delivers a uniform frame, which is appropriate to describe the semantics of state transformations belonging to an imperative programming language as well as the abstract semantics which is induced by a data flow analysis algorithm.

For example, we are able to simulate the complete semantics of a goto programming language w.r.t.:

- an interpretation $I = (D, I_0)$ and
- a set of states Σ_D with data $d \in D$

using the powerset lattice $\mathbf{POT}(\Sigma_D)$ for C and nodes n of the form:

- $n =_{df}$ (boole, instruction)
- $\forall\, \Sigma \subseteq \Sigma_D$: $[\![n]\!](\Sigma) =_{df} \{\sigma \mid \exists\, \sigma \in \Sigma_{boole} \cap \Sigma$: $[\![instruction]\!](\sigma) = \sigma\}$, where Σ_{boole} is the subset of Σ_D leaving the *Boolean* expression *boole* **TRUE**.

In this situation, we are able to translate the characteristic statement of goto programs "if boole then goto L1 else goto L2" in a sub flow graph of the following form:

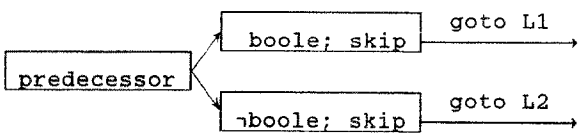

Diagram II.1 .

This concept produces the *collecting semantics* in the sense of Nielson [Ni], which is equivalent to the usual semantics for *flow charts*.

It is also possible to simulate a very general *procedure concept*, even with formal procedure parameters (see sec. **IV**).

The treatment of semantics induced by data flow analysis algorithms follows in the next section.

Finally, we present an alternative *globalization* of a local abstract semantics, which corresponds to the *MFP-solution* of Kam/Ullman [KU].

Definition II.2

Let $G = (N, E, \mathbf{s}, \mathbf{e})$ be a flow graph, $(\mathbb{I}\ \mathbb{I}_l, C)$ a local abstract semantics and $c_s \in C$.

a) A sequence (β_i), $\beta_i : N \longrightarrow C$ is called $(\mathbb{I}\ \mathbb{I}_l, C)$-*computation sequence* of G w.r.t. c_s, if (β_i) fulfils the following properties for every $n \in N$:

i) $$\beta_0(n) = \begin{cases} c_s & , \text{ if } n = \mathbf{s} \\ \top & \text{otherwise,} \end{cases}$$

ii) $\forall\, i \in \mathbb{N}\ \exists\, n_i \in N$:

$$\beta_i(n) = \begin{cases} \left(\bigsqcap_{\substack{\mathbf{n}\ \text{predecessor} \\ \text{of } n}} [\![\mathbf{n}]\!]_l((\beta_{i-1})(\mathbf{n}))\right) \sqcap \beta_{i-1}(n), & \text{if } n = n_i \\ \beta_{i-1}(n) & \text{otherwise.} \end{cases}$$

(β_i) induces a set $NS(\beta_i)$ of node sequences (n_i). We designate the elements of $NS(\beta_i)$ as (β_i)-*node sequences*.

b) Two $(\mathbb{I}\ \mathbb{I}_l, C)$-computation sequences (β_i) and $(\boldsymbol{\beta}_i)$ of G w.r.t. c_s are called *equivalent*, if $\{\beta_0, \beta_1, \ldots\} = \{\boldsymbol{\beta}_0, \boldsymbol{\beta}_1, \ldots\}$.

c) A $(\mathbb{I}\ \mathbb{I}_l, C)$-computation sequence (β_i) of G w.r.t. c_s is called *fair*, if a $(\mathbb{I}\ \mathbb{I}_l, C)$-computation sequence $(\boldsymbol{\beta}_i)$ of G w.r.t. c_s exists, with the following two properties:

- $(\boldsymbol{\beta}_i)$ is equivalent to (β_i).
- a node sequence $(n_i) \in NS(\boldsymbol{\beta}_i)$ exists, so that every node $n \in N$ occurs infinitely often in (n_i).

d) We call $\beta_\infty =_{df} \bigsqcap \beta_i$ (component wise) the *end semantics* of the $(\mathbb{I}\ \mathbb{I}_l, C)$- computation sequence (β_i).

The following theorem states the well-known coincidence of the *MOP*- and *MFP-solution* for distributive frameworks (here the terminology follows [KU], whereas proofs may be found in [CC1, CC2, Ki1 or Ki2]).

The coincidence of these two *globalization concepts* is responsible for the *practical applicability* of the theoretical results about *global abstract semantics* proved in section **III**.

Coincidence Theorem

Let $(\mathbb{I}\ \mathbb{I}, C)$ be the (global) abstract semantics w.r.t. the local abstract semantics $(\mathbb{I}\ \mathbb{I}_l, C)$, $G \in \mathbf{FG}$, $c_s \in C$, (β_i) a fair $(\mathbb{I}\ \mathbb{I}_l, C)$-computation sequence of G w.r.t. c_s and β_∞ the end semantics belonging to (β_i). Then we have:

$$[\![\ G\ ]\!]\ (\ c_s\ ) \quad = \quad \beta_\infty\ (\ \mathbf{e}\ ).$$

Especially:

the end semantics β_∞ is independent of the underlying $(\mathbb{I}\ \mathbb{I}_l, C)$-computation sequence.

III Levels of Abstract Interpretations

During the construction of optimization algorithms questions like
- are two given terms representing the same run time values?
- is a program point reachable?
- does a program part change the values of some special variables?

are of main interest. Using the concept of abstract semantics, we are able to treat such questions on various levels of abstraction. For that purpose, we take two abstract semantics $\mathcal{S}_1 = (\mathbb{I}\ \mathbb{I}_1, C_1)$ and $\mathcal{S}_2 = (\mathbb{I}\ \mathbb{I}_2, C_2)$ and a complete semi-lattice C_3. These three objects characterize three levels of abstraction, which are connected by two distributive and surjective *abstraction functions*:

$A_1 : C_1 \longrightarrow C_2$, with $\forall\, n \in \mathbb{N},\ c \in C_1 : \mathbb{I}n\mathbb{I}_2 \circ A_1(c) \sqsubseteq A_1 \circ \mathbb{I}n\mathbb{I}_1(c)$
and $A_2 : C_2 \longrightarrow C_3$.

To describe the mutual relationship between these levels we additionally introduce the two *adjoint* (or *concretization*) functions $A_1{}^a$ and $A_2{}^a$ defined by:

$\forall\, c \in C_{1+1} : A_1{}^a(c) =_{df} \bigsqcap \{\, \mathbf{c} \mid A_1(\mathbf{c}) = c \,\}$.

Remark III.1

This specifies the following situation. We have a given abstract semantics $\mathcal{S}_1$ and we are interested in some special properties of this semantics (formalized as the elements of C_3), which we want to determine automatically. For that purpose we have to search for an (iteratively) computable abstract semantics $\mathcal{S}_2$ which is detailled enough to completely determine the properties we are interested in (typical examples for such properties are suggested by the three questions mentioned above).

This section delivers criteria to decide whether a given abstract semantics $\mathcal{S}_2$ has this property (i.e. whether $\mathcal{S}_2$ is globally optimal w.r.t. $\mathcal{S}_1$ and C_3 (**Definition III.1.(c)**)) or not.

Remark III.1 motivates the following notations:

- $\mathcal{S}_1$ fixes the *basic level* of our treatment. It stands for the complete semantics of the inspected programming language (compare sec. **II**).
- $\mathcal{S}_2$ characterizes the *computation level.*
- C_3 identifies the *question* (or *observation*) *level.*
- A_1 and A_2 characterize the degree of abstraction between these levels of abstraction.

To illustrate this situation, we use the diagram below:

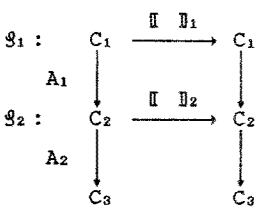

Diagram III.1

Furthermore, we denote the relation between $\mathcal{G}_1$ and C_3 resp. $\mathcal{G}_2$ and C_3 as $\mathcal{G}_1 \longrightarrow_{A_2 \circ A_1} C_3$ resp. $\mathcal{G}_2 \longrightarrow_{A_2} C_3$ and the relation between $\mathcal{G}_1$ and $\mathcal{G}_2$ as $\mathcal{G}_2 \leq_{A_1} \mathcal{G}_1$ or only $\mathcal{G}_2 \leq \mathcal{G}_1$.

Definition III.1

Two abstract semantics $\mathcal{G}_1$ and $\mathcal{G}_2$ are called *isomorph*, iff a distributive and bijective abstraction function $A_1 : \mathcal{G}_1 \longrightarrow \mathcal{G}_2$ exists, fulfilling the following property:

$$\forall\, G \in \mathbf{FG}: \quad [\![G]\!]_2 \circ A_1 = A_1 \circ [\![G]\!]_1 .$$

In this case we write $\mathcal{G}_2 \cong_{A_1} \mathcal{G}_1$ or only $\mathcal{G}_2 \cong \mathcal{G}_1$.

The following lemma collects some immediate results:

Lemma III.1

a) Let $\mathcal{G}_2 \leq_{A_1} \mathcal{G}_1$ and $\mathcal{G}_2 \leq_{A_1} \mathcal{G}_1$. Then we have:

$$(\forall\, c_1, c_2 \in C_1 : A_1(c_1) = A_1(c_2) \implies \mathbf{A_1}(c_1) = \mathbf{A_1}(c_2)) \implies \mathbf{A_1} \circ A_1^a \circ A_1 = \mathbf{A_1} .$$

b) $\mathcal{G}_2 \leq_{A_1} \mathcal{G}_1 \implies$ i) $\forall\, G \in \mathbf{FG}: \quad [\![G]\!]_2 \circ A_1 \subseteq A_1 \circ [\![G]\!]_1$

ii) $\forall\, G \in \mathbf{FG}: \quad [\![G]\!]_2 \subseteq A_1 \circ [\![G]\!]_1 \circ A_1^a$.

c) $\mathcal{G}_2 \leq_{A_1} \mathcal{G}_1 \ \wedge\ \mathcal{G}_1 \leq_{A_1^a} \mathcal{G}_2 \iff \mathcal{G}_2 \cong_{A_1} \mathcal{G}_1$.

Diagram III.1 suggests two different functions to solve the question characterized by C_3:

Firstly, the function optimal for the *basic level*

$$\mathrm{opt} : \mathbf{FG} \times C_3 \longrightarrow C_3 \quad \text{with} \quad \mathrm{opt}(G,c) = A_2 \circ A_1 \circ [\![G]\!]_1 \circ A_1^a \circ A_2^a (c)$$

and secondly, the function given by the *computation level*

$$\mathrm{com} : \mathbf{FG} \times C_3 \longrightarrow C_3 \quad \text{with} \quad \mathrm{com}(G,c) = A_2 \circ [\![G]\!]_2 \circ A_2^a (c).$$

We now want to analyze in which cases $\mathcal{G}_2$ is detailed enough to *simulate* $\mathcal{G}_1$ w.r.t. C_3 (i.e.: $\forall\, G \in \mathbf{FG},\ c \in C_3 : \mathrm{com}(G,c) = \mathrm{opt}(G,c)$).

Definition III.2

a) Let $\mathcal{G}_2 \leq_{A_1} \mathcal{G}_1$ and $\mathcal{G}_2 \longrightarrow_{A_2} C_3$. Then we introduce:

$$RI(\mathcal{G}_1, A_1, \mathcal{G}_2, A_2, C_3) =_{df} \{c \in C_2 \mid \exists\, G \in \mathbf{FG},\ \underline{c} \in C_3 : c = A_1 \circ [\![G]\!]_1 \circ A_1^a \circ A_2^a (\underline{c})\},$$

$RH(\mathcal{G}_1, A_1, \mathcal{G}_2, A_2, C_3)$ denotes the *complete semi-lattice closure* of $RI(\mathcal{G}_1, A_1, \mathcal{G}_2, A_2, C_3)$ and

$RS(\mathcal{G}_1, A_1, \mathcal{G}_2, A_2, C_3) =_{df} ([\![\ ]\!]_2,\ RH(\mathcal{G}_1, A_1, \mathcal{G}_2, A_2, C_3))$, where $[\![\ ]\!]_2$ is defined as follows:

$$\forall\, c \in C_2 : \quad [\![c]\!]_2 =_{df} \begin{cases} [\![c]\!]_2 \mid_{RH(\mathcal{G}_1, A_1, \mathcal{G}_2, A_2, C_3)} & \text{, if } RH(\mathcal{G}_1, A_1, \mathcal{G}_2, A_2, C_3) \text{ is closed under } [\![\ ]\!]_2 \\ \bot & \text{otherwise.} \end{cases}$$

b) C_2 is called *weakly expressive* for $\mathcal{G}_1$ w.r.t. C_3, if the following equation holds for every $G_1, G_2 \in \mathbf{LFG}$:

$$A_2 \circ A_1 \circ [\![G_1]\!]_1 \circ A_1^a \circ A_1 \circ [\![G_2]\!]_1 \circ A_1^a \circ A_2^a = A_2 \circ A_1 \circ [\![G]\!]_1 \circ [\![G_2]\!]_1 \circ A_1^a \circ A_2^a .$$

c) $\mathcal{S}_2$ is called *locally optimal* w.r.t . $\mathcal{S}_1$, if we have:

$\forall\, n \in \mathbf{N}$: $A_1 \circ [\![n]\!]_1 \circ A_1^a = [\![n]\!]_2$.

d) $\mathcal{S}_2$ is called *globally optimal* w.r.t. $\mathcal{S}_1$ and C_3, if $\mathcal{S}_2$ simulates $\mathcal{S}_1$ w.r.t. C_3. In this case, we write $\mathcal{S}_2 \in glob(\mathcal{S}_1, C_3)$.

Remark III.2

a) The trivial "$\perp$"-*case* in **Definition III.2(a)** never occurs in the following considerations.

b) The term *weakly expressive* is suggested by the *Hoare Logic* [Co]. We only have to put $\mathcal{S}_1$ as the semantics induced by an interpretation I and C_2 as the set of those *sets of states* which are characterized by *first order logic formulas*. In this case, we are able to describe the *expressiveness* (of the logic formulas) w.r.t. the programming language **FG** and the interpretation I as follows:

$\forall\, G \in \mathbf{FG},\ c \in C_2$: $[\![G]\!]_1 \circ A_1^a(c) = A_1^a \circ A_1 \circ [\![G]\!]_1 \circ A_1^a(c)$.

That means for every program $G \in \mathbf{FG}$ and every *precondition* $c \in C_2$ a *strongest postcondition* [Co] exists (in C_2), here:

$postcondition(G,c) = A_1 \circ [\![G]\!]_1 \circ A_1^a(c)$.

c) If $\mathcal{S}_2$ is *locally optimal* w.r.t. $\mathcal{S}_1$ then $[\![\ ]\!]_2$ is the *best upper approximation* of $[\![\ ]\!]_1$ in the sense of Cousot/Cousot [CC3].
Thus our concept extends the problem studied by Cousot/Cousot by considering a third level of abstraction (C_3) and by strengthening the optimality requirements.

d) The *global optimality* does not imply the *local optimality* in general, but we will see that this implication holds for every minimal globally optimal abstract semantics (**Equivalence Theorem**).

A first property of the global optimality will be delivered by the

Transitivity Theorem

Let $\mathcal{S}_2 \leq_{A1} \mathcal{S}_2 \leq_{A1} \mathcal{S}_1$. Then we have:

$$\mathcal{S}_2 \in glob(\mathcal{S}_1, C_3) \iff (\mathcal{S}_2 \in glob(\mathcal{S}_1, C_3) \wedge \mathcal{S}_2 \in glob(\mathcal{S}_2, C_3))$$

Proof

"$\Rightarrow$": Let $\mathcal{S}_2 \in glob(\mathcal{S}_1, C_3)$ and $G \in \mathbf{FG}$. In this case we get from **Lemma III.1(b)**:

$$\begin{aligned} A_2 \circ \mathbf{A_1} \circ [\![G]\!]_2 \circ \mathbf{A_1}^a \circ A_2^a &\supseteq A_2 \circ [\![G]\!]_2 \circ A_2^a \\ &= A_2 \circ \mathbf{A_1} \circ A_1 \circ [\![G]\!]_1 \circ A_1^a \circ \mathbf{A_1}^a \circ A_2^a \\ &\supseteq A_2 \circ \mathbf{A_1} \circ [\![G]\!]_2 \circ \mathbf{A_1}^a \circ A_2^a \qquad \text{q.e.d.} \end{aligned}$$

"$\Leftarrow$": Let $\mathcal{S}_2 \in glob(\mathcal{S}_1, C_3)$ and $\mathcal{S}_2 \in glob(\mathcal{S}_2, C_3)$. Then we have for every $G \in \mathbf{FG}$:

$$\begin{aligned} A_2 \circ \mathbf{A_1} \circ A_1 \circ [\![G]\!]_1 \circ A_1^a \circ \mathbf{A_1}^a \circ A_2^a &= A_2 \circ \mathbf{A_1} \circ [\![G]\!]_2 \circ \mathbf{A_1}^a \circ A_2^a \\ &= A_2 \circ [\![G]\!]_2 \circ A_2^a , \end{aligned}$$

and therefore $\mathcal{S}_2 \in glob(\mathcal{S}_1, C_3)$. **q.e.d.**

The local optimality (w.r.t. $\mathcal{G}_1$) is a very natural and easy-to-prove property for an abstract interpretation $\mathcal{G}_2$ which is constructed to simulate $\mathcal{G}_1$. Thus for the (practical) application (in particular for the application in section **IV**) the following characterization of the global optimality is very important:

Simulation Theorem

Let $\mathcal{G}_2$ be a local optimal abstract semantics w.r.t. $\mathcal{G}_1$. Then we have:

$\mathcal{G}_2$ is globally optimal w.r.t. $\mathcal{G}_1$ and C_3 $\iff$

C_2 is weakly expressive for $\mathcal{G}_1$ w.r.t. C_3.

The defining equation for the *global optimality* is too weak for an inductive extension, so that we cannot directly prove the **Simulation Theorem** (by induction). For that reason we introduce a special class of abstract semantics (the *fully abstract models*) to characterize the global optimality. This class is restricted enough to allow easy proofs by induction.

Definition III.2

Let $\mathcal{G}_1$, C and A fulfil $\mathcal{G}_1 \longrightarrow_A C$. $\mathcal{G}_2$ is called a *fully abstract model* of $\mathcal{G}_1$ w.r.t. C and A, if abstraction functions A_1 and A_2 with $\mathcal{G}_2 \leq_{A1} \mathcal{G}_1$, $\mathcal{G}_2 \longrightarrow_{A2} C_3$ and the following four properties exist:

i) $A = A_2 \circ A_1$,

ii) $\mathcal{G}_2 = RS(\mathcal{G}_1, A_1, \mathcal{G}_2, A_2, C)$,

iii) $\mathcal{G}_2$ is locally optimal w.r.t. $RS(\mathcal{G}_1, identity, \mathcal{G}_1, A_2 \circ A_1, C)$,

iv) $\forall\, c_1, c_2 \in RI(\mathcal{G}_1, identity, \mathcal{G}_1, A_2 \circ A_1, C)$:
$A_1(c_1) = A_1(c_2) \iff \forall\, G \in \mathbf{LFG}: A_2 \circ A_1 \circ [\![G]\!]_1(c_1) = A_2 \circ A_1 \circ [\![G]\!]_1(c_2)$.

We denote the set of all those fully abstract models as $FAM(\mathcal{G}_1, A, C)$.

Remark III.3

a) Two data flow informations c_1 and c_2 are generally called *observably equivalent* (w.r.t. C) if $A(c_1) = A(c_2)$. Then condition iv) states that $\mathcal{G}_2$ operates on the *largest congruence* contained in this equivalence. Hence $\mathcal{G}_2$ characterizes a *fully abstract model* in the sense of Milner [Mi].

b) A *fully abstract model* of $\mathcal{G}_1$ w.r.t. C and A depends only on $RS(\mathcal{G}_1, identity, \mathcal{G}_1, A_2 \circ A_1, C)$, C and A. This allows us to assume $\mathcal{G}_1 = RS(\mathcal{G}_1, identity, \mathcal{G}_1, A_2 \circ A_1, C)$ during the following considerations.

Now the distributivity of all the functions mentioned and the "minimality" of the *fully abstract models* deliver:

Lemma III.2

Let $\mathcal{G}_2 \in FAM(\mathcal{G}_1, A_2 \circ A_1, C_3)$ with $\mathcal{G}_2 \leq_{A1} \mathcal{G}_1$. Then the following holds for arbitrary elements c_1, $c_2 \in RH(\mathcal{G}_1, identity, \mathcal{G}_1, A_2 \circ A_1, C_3)$:

$A_1(c_1) = A_1(c_2) \iff \forall\, G \in \mathbf{FG}: A_1 \circ [\![G]\!]_1(c_1) = A_1 \circ [\![G]\!]_1(c_2)$.

Especially, we have as a consequence of $A_1 \circ A_1^a \circ A_1 = A_1$:

$\forall\, G_1, G_2 \in \mathbf{FG}: A_1 \circ [\![G_1]\!]_1 \circ [\![G_2]\!]_1(c_1) = A_1 \circ [\![G_1]\!]_1 \circ A_1^a \circ A_1 \circ [\![G_2]\!]_1(c_1)$.

This enables us to state the

Existence and Uniqueness Theorem

Let $\mathcal{S}_1$, C and A fulfil $\mathcal{S}_1 \longrightarrow_A C$. Then (with the exception of isomorphism "$\cong$") a unique *fully abstract model* of $\mathcal{S}_1$ w.r.t. C and A exists.

Proof

To prove the existence, we only have to consider the local optimal abstract interpretation induced on the *largest congruence* by $RS(\mathcal{S}_1, \text{identity}, \mathcal{S}_1, A_2 \circ A_1, C)$.

The uniqueness will be derived by using **Lemma III.1(a and b)**, **Lemma III.2** and **Remark.III.2(b)**. These lemmata deliver the right hand side condition of **Lemma III.1.(c)**. **q.e.d.**

The next theorem is a consequence of **Lemma III.1**. It can be proved by an easy induction argument.

Theorem III.1

Let $\mathcal{S}_2 \in FAM(\mathcal{S}_1, A_2 \circ A_1, C_3)$ with $\mathcal{S}_2 \leq_{A_1} \mathcal{S}_1$. Then we have:

$\forall\, G \in \mathbf{FG}$: $A_1 \circ [\![G]\!]_1 \circ A_1^a \circ A_2^a = [\![G]\!]_2 \circ A_2^a$.

In particular: $\mathcal{S}_2$ is globally optimal w.r.t. $\mathcal{S}_1$ und C_3.

The main part of the argumentation for the **Simulation Theorem** consists of verifying the following theorem:

Theorem III.2

Let $\mathcal{S}_1 \longrightarrow_{A \circ A_1} C_3$, $\mathcal{S}_2$ ($\leq_{A_1} \mathcal{S}_1$) be locally optimal w.r.t. $\mathcal{S}_1$, C_2 be weakly expressive for $\mathcal{S}_1$ w.r.t. C_3 and $\mathbf{\mathcal{S}_2} \in FAM(\mathcal{S}_2, A, C_3)$. In this case, $\mathbf{\mathcal{S}_2}$ is a *fully abstract model* of $\mathcal{S}_1$ w.r.t. C_3 and $A \circ A_1$.

To prove this result, we consider the following situation:

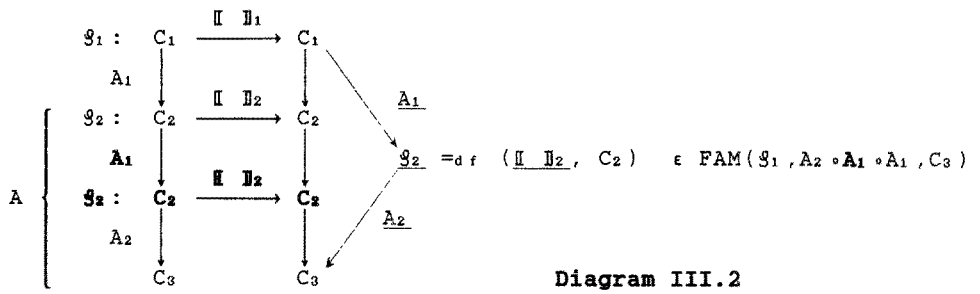

Diagram III.2

Here the idea of the argument needed consists of deriving the left hand side condition of **Lemma III.1(c)** for $\mathcal{S}_2$ and $\mathbf{\mathcal{S}_2}$. Thus we may conclude $\mathcal{S}_2 \cong \mathbf{\mathcal{S}_2}$ to obtain the assertion stated (a complete proof is presented in [St]).

After this preparation we are able to prove the main theorem of this section. We produce this proof in more detail to give an insight into some typical argumentations in this context.

Equivalence Theorem

Let $\mathcal{S}_2 \leq_{A_1} \mathcal{S}_1$ and $\mathcal{S}_1 \longrightarrow_{A \circ A_1} C_3$. Then we have:

$\mathcal{S}_2 \in glob(\mathcal{S}_1, C_3) \iff FAM(\mathcal{S}_2, A, C_3) = FAM(\mathcal{S}_1, A \circ A_1, C_3)$

Proof

"$\Leftarrow$" follows immediately by applying **Theorem III.1** twice.

To prove the converse let $\mathcal{G}_2 \in glob(\mathcal{G}_1, C_3)$, $A = A_2 \circ \mathbf{A_1}$ and $\mathcal{G}_2 \leq_{A1} \mathcal{G}_2$ with $\mathcal{G}_2 = (\llbracket \mathbf{I}_2, \mathbf{C_2})$ $\in FAM(\mathcal{G}_2, A_2 \circ \mathbf{A_1}, C_3)$. Then we have for arbitrary flow graphs G_1, $G_2 \in \mathbf{FG}$:

$A_2 \circ \mathbf{A_1} \circ A_1 \circ \llbracket G_1 \rrbracket_1 \circ \llbracket G_2 \rrbracket_1 \circ A_1{}^a \circ \mathbf{A_1}{}^a \circ A_2{}^a$ = (glob. opt.)

$A_2 \circ \mathbf{A_1} \circ \llbracket G_1 \rrbracket_2 \circ \llbracket G_2 \rrbracket_2 \circ \mathbf{A_1}{}^a \circ A_2{}^a$ = (**Theo. III.1**)

$A_2 \circ \llbracket G_1 \rrbracket_2 \circ \llbracket G_2 \rrbracket_2 \circ A_2{}^a$ $\sqsubseteq$ (**Lemma III.1(b)**)

1) $A_2 \circ \mathbf{A_1} \circ \llbracket G_1 \rrbracket_2 \circ \mathbf{A_1}{}^a \circ \llbracket G_2 \rrbracket_2 \circ A_2{}^a$ $\sqsubseteq$ (")

2) $A_2 \circ \mathbf{A_1} \circ \llbracket G_1 \rrbracket_2 \circ \mathbf{A_1}{}^a \circ \mathbf{A_1} \circ A_1 \circ \llbracket G_2 \rrbracket_1 \circ A_1{}^a \circ \mathbf{A_1}{}^a \circ A_2{}^a$ $\sqsubseteq$ (")

3) $A_2 \circ \mathbf{A_1} \circ A_1 \circ \llbracket G_1 \rrbracket_1 \circ A_1{}^a \circ \mathbf{A_1}{}^a \circ \mathbf{A_1} \circ A_1 \circ \llbracket G_2 \rrbracket_1 \circ A_1{}^a \circ \mathbf{A_1}{}^a \circ A_2{}^a$ $\sqsubseteq$

4) $A_2 \circ \mathbf{A_1} \circ A_1 \circ \llbracket G_1 \rrbracket_1 \circ \llbracket G_2 \rrbracket_1 \circ A_1{}^a \circ \mathbf{A_1}{}^a \circ A_2{}^a$.

This requires the equality in all cases. Therefore we obtain the weak expressiveness of $\mathbf{C_2}$ w.r.t. $\mathcal{G}_1$ by 3) and 4). Furthermore we get from 1) and 2) because of **Definition III.2(iv)**:

$\forall\, G \in \mathbf{FG}$: $\mathbf{A_1} \circ \mathbf{A_1}{}^a \circ \llbracket G \rrbracket_2 \circ A_2{}^a = \mathbf{A_1} \circ A_1 \circ \llbracket G \rrbracket_1 \circ A_1{}^a \circ \mathbf{A_1}{}^a \circ A_2{}^a$.

This obviously yields the local optimality of $\mathcal{G}_2$ w.r.t. $\mathcal{G}_1$ (notice **Remark III.3.(b)**). Together with **Theorem III.2** we obtain that $\mathcal{G}_2$ ($\in FAM(\mathcal{G}_2, A_2 \circ \mathbf{A_1}, C_3)$) is a *fully abstract model* of $\mathcal{G}_1$ w.r.t. C_3 and $A_2 \circ \mathbf{A_1} \circ A_1$.

Now the **Existence and Uniqueness Theorem** delivers the assertion stated. **q.e.d.**

The **Simulation Theorem** (which delivers the *optimality* of the *first analysis stage* (sec. **IV**)) is clearly an immediate consequence of the last two theorems.

Our concept of the *three levels of abstract interpretations* delivers a uniform frame for the classification of iterative analysis algorithms. To illustrate the meaning of these three levels, we finish with an example which is concentrated on this phenomenon.

Example:

Let us have a look at the following concretization of the general situation:

- $\mathbf{N} = \{ n_1, n_2, n_3 \}$,
- $\mathcal{G}_1 = (\llbracket \mathbf{I}_1, C_1)$ with:
  - $C_1 = \{ M \mid M \subseteq \mathbb{N} \}$,
  - $\llbracket n_1 \rrbracket_1$ = "x:= x+1", $\llbracket n_2 \rrbracket_1$ = "x:= 2", $\llbracket n_3 \rrbracket_1$ = "x:= x*3",
- $C_3 = \{ \bot, 3|x, \neg(3|x), \top \}$,
- A is the abstraction function belonging to C_1 and C_3.

This situation is suitable for studying the question whether the value of x is divisible by 3.

In this section we have treated such situations to obtain criteria for an abstract semantics $\mathcal{G}_2$ to prove whether $\mathcal{G}_2$ is globally optimal w.r.t. $\mathcal{G}_1$ and C_3. This means here, whether $\mathcal{G}_2$ solves the question of the divisibility of x by 3.

In our case, it is sufficient to compute *modulo 3* . That leads to the following choice of $\mathcal{S}_2$:

$\mathcal{S}_2 = (\mathbb{I}\ \mathbb{I}_2, C_2)$ with $\mathbb{I}n_1\ \mathbb{I}_2 =_{df} \mathbb{I}n_1\ \mathbb{I}_1$ *"mod 3"* and

$C_2 =_{df}$

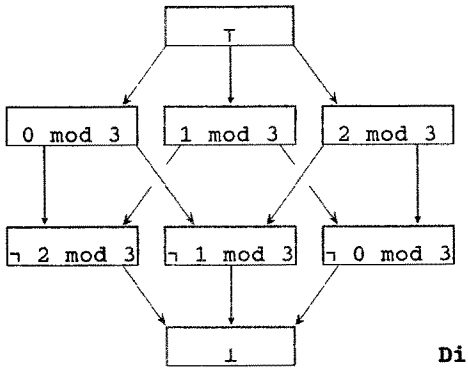

Diagram III.3

It is easy to show that $\mathcal{S}_2$ is even a *fully abstract model* of $\mathcal{S}_1$ w.r.t. C_3 and A.

Remark III.3

Small changes to **N**, $\mathcal{S}_1$ and C_3 lead to substantial alterations to $\mathcal{S}_2$. For example, allowing a statement of the form "if x = 7 then x:= 3" raises the complexity of a global optimal abstract semantics enormously.

IV The First Analysis Stage

The data flow analysis algorithm implemented by us consists as usual of two components:

1) A local data flow analysis process to analyze elementary statements or *basic blocks* [He]. This process defines a local abstract semantics $(\mathbb{I}\ \mathbb{I}_1, C)$.
2) A *scheduler* realizing fairness by means of a *worklist* [Ki1, Ki2 or He]. It provides for the iterative globalization of the local data flow process generating fair $(\mathbb{I}\ \mathbb{I}_1, C)$-computation sequences only.

This algorithm specifies a process for successively constructing a fair $(\mathbb{I}\ \mathbb{I}_1, C)$-computation sequence. Using the **Compositionality Theorem** (sec. **II**), we learn that this process approximates the global abstract semantics $\mathcal{S}_2$ belonging to $(\mathbb{I}\ \mathbb{I}_1, C)$. To prove the *Herbrand optimality* of this process, we only have to show that:

- the *end semantics* characterizing $\mathcal{S}_2$ is reached in a finite number of steps
- $\mathcal{S}_2$ is *Herbrand optimal*.

Let therefore: - **T** be the *Herbrand universe* determined by the variables and operators of the underlying programming language,

- Part(T) be the set of all partitions over $T \subseteq \mathbf{T}$,
- $\mathbf{Part(T)} =_{df} \{\ p \mid \exists\ T \subseteq \mathbf{T}\colon p \in \mathrm{Part}(T) \wedge |T| \in \mathbb{N}\ \}$.

Then we have to analyze the following situation:

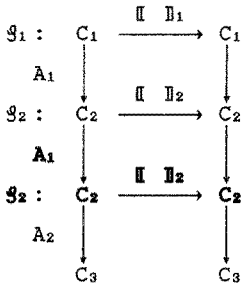

Diagram IV.1

Here: - Let $\mathfrak{S}_1$ be the abstract semantics induced by the *Herbrand interpretation* (In this case, C_1 is the powerset lattice of the set of all *Herbrand states* with "$\sqcap$" $=_{df}$ "$\cup$").

- Let $A_1 : C_1 \longrightarrow \mathrm{Part}(\mathbf{T})$ be the function, which relates the partition $p \in \mathrm{Part}(\mathbf{T})$ to any set of states $\Sigma \in C_1$. Partition p reduces the detailed (data flow) information given by Σ to the viewpoint:

  value equality between program terms.

- $\mathfrak{S}_2 =_{df} (\llbracket\ \rrbracket_2, C_2)$, with:
  - $C_2 =_{df} \{ p \in \mathrm{Part}(\mathbf{T}) \mid \exists\, \Sigma \in C_1 : A_1(\Sigma) = p \}$ and
  - $\llbracket\ \rrbracket_2$ being the globalization of:
    $\llbracket\ \rrbracket_{2\,1} : \mathbf{N} \longrightarrow (C_2 \longrightarrow C_2)$; $\forall\, n \in \mathbf{N}: \llbracket n \rrbracket_{2\,1} = A_1 \circ \llbracket n \rrbracket_1 \circ A_1^{a}$.

- Let $\mathbf{A_1} : C_2 \longrightarrow \mathbf{Part}(\mathbf{T})$ be the partially defined function which relates to any partition $p \in C_2$ belonging to a set of *Herbrand states* $\Sigma \in C_1$ the smallest partition $\mathbf{p} \in \mathbf{Part}(\mathbf{T})$ describing Σ, if such a partition exists.

- Let $\mathfrak{S}_2$ be formed in analogy to $\mathfrak{S}_2$.

Applying the **Simulation Theorem** (sec. **III**) we are able to prove:

- $\mathfrak{S}_2$ is globally optimal w.r.t. $\mathfrak{S}_1$ and C_3 and
- $\mathfrak{S}_2$ is globally optimal w.r.t. $\mathfrak{S}_2$ and C_3.

Together with the **Transitivity Theorem** (sec. **III**) this yields the global optimality of $\mathfrak{S}_2$ w.r.t. $\mathfrak{S}_1$ and C_3.

On the other hand, it is possible to show that $\mathbf{C_2}$ fulfils the *descending chain condition* (i.e.: for every $\mathbf{C_2}$-sequence, we have:

$$(\forall\, i \in \mathbb{N}: c_i \sqsupseteq c_{i+1}) \Longrightarrow (\exists\, i \in \mathbb{N}\ \forall\, j \in \mathbb{N}: c_i = c_{i+j})).$$

Combining these two properties we get:

Optimality Theorem

The implemented algorithm computes exactly the partition of $\mathbf{C_2}$, which is optimal to describe the set of *Herbrand states* related to a given program point.

Alternatively:

The algorithm determines all *value equalities* between program terms for every program point valid for every interpretation.

Remark IV.1

This theorem does not hold for the analysis algorithms occurring in the literature. For example most of the *computational levels* considered there were not even *locally optimal* w.r.t. the *Herbrand interpretation*.

As stated in section **II**, our approach is qualified for the analysis of arbitrary *goto programs*, but this does not limit its applicability. Using an additional process, (similar to our standard process), we are able to extend the analysis to an *interprocedural level*. By this means, it is possible to reach the *Herbrand optimality* even if the source program is provided with *formal procedure parameters* (to prove this, it has to be argued by means of the **Simulation** and **Transitivity Theorem** again).

The additional process is based on the following idea:

We lift the *functionality* of our standard process. That means, instead of C, we use $\mathbf{C} =_{df} \{ Z \mid Z: C \longrightarrow C \}$ as the complete semi-lattice of data flow informations. To do this, we have to change our semantic function $[\![\ ]\!]$ as follows:

$$[\![\ ]\!]: \mathbf{N} \longrightarrow (C \longrightarrow C) \quad \text{with} \quad [\![n]\!](Z) = [\![n]\!] \circ Z.$$

By this means, we are able to use our standard process to approximate the complete *state transformation* of a procedure. This approximation process is superposed by a *control iteration* distributing the approximate *state transformations* of the distinct procedures.

Remark IV.2

a) **C** does not fulfil the descending chain condition. Nevertheless we are able to prove the *termination* of our *interprocedural process*, because a given program G only allows very special *descending chains*.

Indeed an effective construction process exists which delivers a finite set of terms T(G), so that Part(T(G)) × Part(T(G)) is powerful enough to express all the elements of **C** which possibly occur during an interprocedural analysis of G.

Thus our algorithm preserves its optimality even in the *interprocedural* context.

b) The concept of the *superposed control iterations* leads to a *Herbrand optimal* treatment of *local variables*. This property distinguishes it from the process published by Barth [Ba], where it is unclear how to handle local variables of recursive procedures at all.

c) Our process allows *first order procedures* only. To remove this limitation, we have to use a preprocess which transforms programs with *higher order procedures* into those with *first order procedures* only.

Such a preprocess exists whenever the given program G possesses a *regular formal call tree* [Ol] or more specificially, whenever G possesses no *global formal procedure parameters* [La].

Langmaack [La] resp. Olderog [Ol] has shown (by using the technique of *accompanying parameters* [La]) that such programs can effectively be transformed into equivalent ones without any *procedure*

nesting. Subsequently the resulting programs can be easily transformed into programs with *first order procedures* only, which are *equivalent* to the programs we have started with.

Thus it is possible to obtain *Herbrand optimal* results for the whole class of those programs which possess a *regular formal call tree*.

V. The Second Analysis Stage

The second analysis stage is also based on the globalization of a local abstract semantics of a given flow graph G, but here we use *bit vectors* representing the truth value of some predicates instead of partitions over a set of terms. Thus globalization means solving a *Boolean equation system* induced by these predicates. The solutions of this equation system deliver the (optimal) *computation points* belonging to special *systems of value equivalence classes* (these systems represent an *abstract value*, the computations of which we intend to *locate* optimally within the flow graph G).

The whole *location process* (for one *abstract value*) is organized in six steps:

1) Insert a node into each edge of G (these artificial nodes mark the potential computation points of the distinct systems of value equivalence classes).

2) Use the *first analysis stage* to determine for every node n of G a *pre-* and *postpartition* to describe the *value equivalences* between the program terms at these program points of G.

3) Construct the *value flow graph* VFG = (VN,VE). Nodes **n** $\in$ VN are the equivalence classes of the *pre-* and *postpartitions* mentioned above. Edges **e** $\in$ VE are the pairs (V,W), where V resp. W is a equivalence class of the *pre-* resp. *postpartition* of a node n resp. m of G, fulfilling one of the following two properties:

   - n = m and whenever an element v $\in$ V has a special value x before the execution of n, then every element w $\in$ W has the same value x after the execution of n (this relation is also to be determined by the *first analysis stage*).
   - n is a predecessor of m and V $\supseteq$ W.

4) Determine the *systems of value equivalence classes* which are characterized as the *maximal connected subgraphs* ($\underline{N}$,$\underline{E}$) of VFG with:
   for all *pre-* and *postpartitions* V determined in step 2:
   $|\ \underline{N} \cap \{\ K \mid K \text{ is an equivalence class of } V\ \}\ | \leq 1$.

   Each system of this kind serves for a special initialization of the *Boolean equation system* mentioned above.

5) Choose such a *system of equivalence classes* to initialize the *Boolean equation system*.

6) Solve the (already intialized) *Boolean equation system* to obtain the *optimal computation points*.

Remark V.1

This approach is based on the method stated by Morel/Renvoise [MR], which is conceived only to treat the occurrences of a single given term. Our concept of the *value flow graph* leads to an extension of this method which solves the *location problem* for a whole class of *Herbrand equivalent* terms (i.e. it determines the best *computation points* which are valid for every interpretation).

After having determined the *optimal computation points* in this way, we receive the *optimal computation forms* as minimal representatives of the value equivalence classes being related to the computation points.

VI. The Optimization

Having finished the *analysis stage*, we are able to optimize our program G for every system of *value equivalence classes* by the insertion of an auxiliary variable **h** as follows:

- Firstly, we initialize **h** at every *computation point* by means of a *computation form* being valid at this point.
- Secondly, we substitute every occurrence of a term being related to the inspected system by **h**.

These optimizations induced by the various systems of *value equivalence classes* are largely independent. This allows us to determine the computation forms and points completely in parallel. Only the program transformations make some difficulties because the various systems of value equivalence classes overlap. For that reason, it is advantageous to do all the initilizations in parallel, before going on to substitute the program terms using the auxiliary variables. (It is best to base these substitutions on the results of an extra run of the first analysis stage to eliminate all the redundancies, even those occurring between the terms we just have inserted.)

This procedure transforms a given program G ε **FG** into a minimal form **G** w.r.t. the *Herbrand interpretation* and the branching structure of G:

Every path of **G** contains a minimal number of *Herbrand equivalent* computations w.r.t. to the branching structure of G.

That means that the expenditure of the computations of **G** is only reducible by changing the branching structure or by utilizing special properties of the underlying interpretation.

Our approach is not qualified to change the branching structure of a program, but it supports the utilization of special properties of a given interpretation.

VII. Literature

[Ba] Barth, G. *"Interprozedurale Datenflußsysteme"*, Habilitationsschrift, Universität Kaiserslautern, 1981

[CC1] Cousot, P. and Cousot, R. *"Abstract interpretation: A unified Lattice Model for static Analysis of Programs by Construction or Approximation of Fixpoints"*, 4th POPL, Los Angeles, California, 238 - 252, 1977

[CC2] Cousot, P. and Cousot, R. *"Automatic Synthesis of Optimal Invariant Assertations: Mathematical Foundations"*, ACM Sigplan Notices 12, 1 - 12, 1977

[CC3] Cousot, P. and Cousot, R. *"Systematic Design of Program Analysis Frameworks"*, 6th POPL, San Antonio, Texas, 269 -282, 1979

[Co] Cook, S. A. *"Soundness and Completeness of an Axiom System for Program Verification"*, SIAM Journal Computing, 7 : 1, 70 - 90, 1978

[FKU] Fong, A. C., Kam, J. B. and Ullman, J. D. *"Applications of Lattice Algebra to Loop Optimization"*, 2nd POPL, Palo Alto, California, 1 - 9, 1975

[Gr] Greibach, S. A. *"Theory of Program Structures: Schemes, Semantics, Verification"*, LNCS 36, Springer-Verlag, 1975

[He] Hecht, M. S. *"Flow Analysis of Computer Programs"*, Elsevier, North-Holland, 1977

[KU] Kam, J. B. and Ullman, J. D. *"Monotone Data Flow Analysis Frameworks"*, Acta Informatica, 7, 309 - 317, 1975

[Ki1] Kildall, G. A. *"Global Expression Optimization during Compilation"*, Technical Report No. 72-06-02, University of Washington, Computer Science Group, Seattle, Washington, 1972

[Ki2] Kildall, G. A. *"A Unified Approach to Global Program Optimization"*, 1st POPL, Boston, Massachusetts, 194 - 206, 1973

[La] Langmaack, H. *"On Procedures as Open Subroutines.* I " Acta Informatica 2, 311 - 333, 1973

[Mi] Milner, R. *"Fully Abstract Models of Typed λ-Calculi"*, TCS 4, North Holland, 1 - 22, 1977

[MR] Morel, E. and Renvoise, C. *"Global Optimization by Suppression of Partial Redundancies"*, CACM, 22 : 2, 96 - 103, 1979

[Ni] Nielson, F. *"A Bibliography on Abstract Interpretations"*, ACM Sigplan Notices 21, 31 - 38, 1986

[Ol] Olderog, E.-R. *"Charakterisierung Hoarescher Systeme für ALGOL - ähnliche Programmiersprachen"*, Dissertation, Christian-Albrechts-Universität Kiel, 1981

[RL] Reif, J. H. and Lewis, R. *"Symbolic Evaluation and the Global Value Graph"* , 4th POPL, Los Angeles, California, 238 - 252, 1977

[St] Steffen, B., doctoral dissertation, to appear, 1987