Background, Reserve, and Gandy Machines

Andreas Blass[1,⋆] and Yuri Gurevich[2]

[1] Mathematics Dept., University of Michigan, Ann Arbor, MI 48109–1109, U.S.A.
ablass@umich.edu
[2] Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
gurevich@microsoft.com

Abstract. The reserve of a state of an abstract state machine was defined to be a "naked set". In applications, it may be convenient to have tuples, sets, lists, arrays, etc. defined ahead of time on all elements, including the reserve elements. We generalize the notion of reserve appropriately. As an application, we solve a foundational problem in Gandy's formalization of mechanical devices.

Part 1
Introduction and Preliminaries

1 Introduction

In this paper, we address two closely related foundational issues. We encountered these issues in connection with the notion of "reserve" in abstract state machines (ASMs), but they are of somewhat broader relevance, for example to the computing mechanisms described in [Gandy 1980]. In this introduction, we shall discuss these issues in general terms, specializing later to the cases of primary interest.

Algorithms often need to increase their working space, and there are two ways to view this increase. One is that the additional space was really there all along but was not previously used; the other is that genuinely new space is created. For example, from the first viewpoint, a Turing machine has an infinite tape, only finitely much of which is in use at any stage of the computation. From the second viewpoint, a Turing machine's tape is finite but new squares can be appended to it as needed.

For foundational purposes, it is usually more convenient to adopt the first viewpoint, so as not to have to worry about the nature of newly created elements. In

⋆ Preparation of this paper was partially supported by a grant from Microsoft Corporation.

P. Clote and H. Schwichtenberg (Eds.): CSL 2000, LNCS 1862, pp. 1–17, 2000.

particular, ASMs have, by definition [Gurevich 1995], an infinite reserve which is a part of the base set. All the basic functions and relations except equality take only their default values if at least one of the arguments belongs to the reserve, and no basic function outputs a reserve element. When a new element is needed in the active part of the state, one is imported from the reserve.

Although the reserve has no internal structure (except equality), it is often desirable to have some external structure over it. For example, in [BGS 1999] every state was required to include all the hereditarily finite sets over its atoms. This means that finite sets of reserve elements (and finite sets of these, etc.) were present, with their membership relation. Thus, when a reserve element is imported, sets involving it already exist and do not need to be created separately (by importing additional elements and appropriately defining the membership relation on them). Similarly, one might want to have other sorts of structure already available, for example lists or arrays of reserve elements.

The first issue treated in this paper is to make precise the notion of a sort of structure (like sets, or lists, or arrays) that can exist above a set of atoms without putting any structure (except equality) on the atoms themselves. We formalize this in the notion of a background class of structures. Thus, for example, the background class relevant to [BGS 1999] would consist of structures of the form: set U of atoms plus all hereditarily finite sets over U (as well as isomorphic copies of such structures). The idea is that such a class of structures specifies the constructions (like finite sets) available as "background" for algorithms.

The second issue is the choice of elements to be imported from the reserve. If the importation is to be algorithmic, it must be non-deterministic, since an algorithm has no way to distinguish one reserve element from another. But this sort of non-determinism is intuitively much more benign than general non-determinism. We attempt to capture what accounts for this intuition, by introducing the notion of inessential non-determinism. The idea here is that the various options allowed by the non-determinism all lead to isomorphic states, so that it makes no difference which option is chosen.

Alternatively, one could insist on determinism, specifying a particular one of the available reserve elements to be imported. This is the approach used in [Gandy 1980]. The price of this insistence is that the specification cannot be algorithmic or even canonical. We shall show how to turn a Gandy-style deterministic, non-algorithmic process into a non-deterministic algorithm of the sort described above, and we shall prove that Gandy's notion of "structural" for his processes corresponds to our notion of "inessential non-determinism."

2 Structures

The notion of (first-order) structure is found in textbooks on mathematical logic; for example see [Enderton 1972]. We use a slight modification of the notion of classical structures [Gurevich 1991].

2.1 Syntax

A *vocabulary* is a finite collection of function names, each of a fixed arity. Some function names may be marked as *relational*. Every vocabulary contains the equality sign, the nullary names `true`, `false`, `undef`, the unary name `Boole`, and the names of the usual Boolean operations. With the exception of `undef`, all these *logic names* are relational. A function name can be marked (by the vocabulary) as *static*.

Terms (more exactly *ground terms*; by default, terms are ground in this article) are defined by the usual induction. A nullary function name is a term. If f is a function name of positive arity j and if $t_1, \ldots, t_j$ are terms, then $f(t_1, \ldots, t_j)$ is a term. If the outermost function name is relational, then the term is *Boolean*.

2.2 Semantics

A *structure* X of vocabulary Υ is a nonempty set S (the *base set* of X) together with interpretations of the function names in Υ over S. Elements of S are also called elements of X. A j-ary function name is interpreted as a function from S^j to S, a *basic function* of X. We identify a nullary function with its value. Thus, in the context of a given structure, `true` means a particular element, namely the interpretation of the name `true`; the same applies to `false` and `undef`. It is required that `true` be distinct from the interpretations of the names `false` and `undef`. The interpretation of a j-ary relation R is a function from S^j to $\{\texttt{true}, \texttt{false}\}$, a *basic relation* of X. The equality sign is interpreted as the identity relation on the base set. Think about a basic relation R as the set of tuples $\bar{a}$ such that $R(\bar{a}) = \texttt{true}$. If relation R is unary it can be viewed as a *universe*. `Boole` is (interpreted as) the universe $\{\texttt{true}, \texttt{false}\}$. The Boolean operations behave in the usual way on `Boole` and produce `false` if at least one of the arguments is not Boolean. `undef` allows us to represent intuitively-partial functions as total.

The *domain* of a non-relational j-ary basic function f is the set of j-tuples $\bar{a}$ such that $f(\bar{a}) \neq \texttt{undef}$. The *range* of f is the set of elements $f(\bar{a})$ different from `undef` where $\bar{a}$ ranges over all j-tuples of elements of the base set.

A straightforward induction gives the value $Val(t, X)$ of a term t in a structure X whose vocabulary includes that of t. If $Val(t, X) = Val(t', X)$, we may say that $t = t'$ in X. If $t = \texttt{true}$ (resp. $t = \texttt{false}$) in X, we may say that t holds or is true (resp. fails or is false) in X.

3 Sequential-Time and Abstract-State Postulates

Restrict attention to one-thread algorithms. (In the terminology of [Gurevich 1995], this means that we allow parallel algorithms but not independent agents.) Following [Gurevich 2000], we make the following assumptions.

Sequential Time

Postulate 1 (Deterministic Sequential Time) Every deterministic algorithm A is associated with

- a set $\mathcal{S}(A)$ whose elements will be called *states* of A,
- a subset $\mathcal{I}(A)$ of $\mathcal{S}(A)$ whose elements will be called *initial states* of A, and
- a map $\tau_A : \mathcal{S}(A) \longrightarrow \mathcal{S}(A)$ that will be called the *one-step transformation* of A.

Postulate 2 (Nondeterministic Sequential Time) Every nondeterministic algorithm A is associated with

- a set $\mathcal{S}(A)$ whose elements will be called *states* of A,
- a subset $\mathcal{I}(A)$ of $\mathcal{S}(A)$ whose elements will be called *initial states* of A, and
- a relation $\tau_A \subseteq \mathcal{S}(A) \times \mathcal{S}(A)$ that will be called the *one-step transformation* of A.

Abstract State

Postulate 3 (Deterministic Abstract State) Let A be an arbitrary deterministic algorithm.

- States of A are first-order structures.
- All states of A have the same vocabulary.
- The one-step transformation τ_A does not change the base set of any state. Nor does it change the interpretations of static basic functions.
- $\mathcal{S}(A)$ and $\mathcal{I}(A)$ are closed under isomorphisms. Further, any isomorphism from a state X onto a state Y is also an isomorphism from $\tau_A(X)$ onto $\tau_A(Y)$.

Postulate 4 (Nondeterministic Abstract State) Let A be an arbitrary nondeterministic algorithm.

- States of A are first-order structures.
- All states of A have the same vocabulary.
- If $(X, X') \in \tau_A$ then X and X' have the same base set and the same basic static functions.
- $\mathcal{S}(A)$ and $\mathcal{I}(A)$ are closed under isomorphisms. Further, let ζ be an isomorphism from a state X onto a state Y. For every state X' with $(X, X') \in \tau_A$, there is a state Y' with $(Y, Y') \in \tau_A$ such that ζ is an isomorphism from X' onto Y'.

Notation. $\tau[A] \rightleftharpoons \tau_A$. Notation $\tau[A]$ is more convenient when the algorithm name is complex.

Lemma 3.1 (Symmetry Preservation) *Suppose that A is a deterministic algorithm and let $X \in \mathcal{S}(A)$. Every automorphism of X is also an automorphism of $\tau_A(X)$.*

Proof This is the last part of the deterministic abstract-state postulate applied to the special case that $X = Y$. □

3.1 Hereditarily Finite Sets

We recall a couple of set-theoretic notions.

A set x is *transitive* if it contains all elements of its elements: if $z \in y \in x$ then $z \in x$. The *transitive closure* $\mathrm{TC}(x)$ of a set x is the least transitive set that includes x. A set x is *hereditarily finite* if $\mathrm{TC}(x)$ is finite.

A set x may contain elements which are not sets; these are called *atoms*. The collection of atoms in $\mathrm{TC}(x)$ is the *atomic support* of x. Following Gandy [1980], the atomic support of a set x will be denoted $\mathrm{Sup}(x)$.

Let U be a set of atoms. The collection $\mathrm{HF}(U)$ of *hereditarily finite sets over U* is the collection of all hereditarily finite sets x with $\mathrm{Sup}(x) \subseteq U$. It is easy to see that $\mathrm{HF}(U)$ is the least set S such that every finite subset of $U \cup S$ is an element of S.

Corollary 3.2 *Consider a family $\{U_i : i \in I\}$ of subsets of U. We have*

$$\bigcap_i HF(U_i) = HF\Big(\bigcap_i U_i\Big)$$

Proof By definition of HF and of intersection, a set x belongs to $\bigcap_i \mathrm{HF}(U_i)$ if and only if $\mathrm{Sup}(x) \subseteq U_i$ for each i. But this is the same as saying $\mathrm{Sup}(x) \subseteq \bigcap_i U_i$, which is the definition of x belonging to $\mathrm{HF}\Big(\bigcap_i U_i\Big)$. □

Part 2
Background Classes and Reserve

4 Background Classes

4.1 Preliminaries

In this section, every vocabulary contains a unary predicate Atomic. We call this predicate and the logical symbols *obligatory* and all other symbols *non-obligatory*. If $X \models \mathrm{Atomic}(x)$, call x an *atom* of X. The set of atoms of X will

be denoted Atoms(X). X is *explicitly atom-generated* if the smallest substructure of X that includes all atoms is X itself.

Given two structures X, Y of the same vocabulary, we write $X \leq Y$ to indicate that X is a substructure of Y. If $X \leq Y$ and X belongs to a class K of structures then X is a *K-substructure* of Y.

4.2 Main Definitions

Definition 4.1 A class K of structures of a fixed vocabulary is a *background class* if the following requirements BC0–BC3 are satisfied.

BC0 K is closed under isomorphisms.
BC1 For every set U, there is a structure $X \in K$ with Atoms(X) $= U$.
BC2 For all $X, Y \in K$ and every embedding (of sets) ζ : Atoms(X) $\to$ Atoms(Y), there is a unique embedding (of structures) η of X into Y that extends ζ.
BC3 For all $X \in K$ and every $x \in$ Base(X), there is a smallest K-substructure Y of X that contains x.

Definition 4.2 Suppose that K is a background class, $X \in K$, $S \subseteq$ Base(X), and F is the set of substructures $Y \leq X$ such that Y belongs to K and includes S. If F has a smallest member Y then Y is called the *envelope* $E_X(S)$ of S in X and Atoms(Y) is called the *support* $\mathrm{Sup}_X(S)$ of S in X.

Notice that the smallest background substructure of X that contains a particular element $x \in X$ is $E_X(\{x\})$. It is tempting to simplify the notation by writing simply $E_X(x)$, but this can lead to ambiguity if x is both an element and a subset of X.

BC3 asserts that, in a background structure, every singleton subset has an envelope.

Definition 4.3 *A background class K is* finitary *if, in every background structure, the support of every singleton set is finite.*

4.3 Analysis

Let K be a background class. Members of K are *background structures*, K-substructures are *background substructures*.

Lemma 4.4 *In BC2, if ζ is onto then η is onto as well.*

Proof Suppose that ζ is onto. By BC2 (existence), ζ^{-1} extends to an embedding $\theta : Y \to X$. The identity map $\zeta \circ \zeta^{-1}$: Atoms(Y) $\to$ Atoms(Y) extends to $\eta \circ \theta : Y \to Y$. By BC2 (uniqueness), $\eta \circ \theta$ is the identity map on Y. It follows that η is onto. □

Lemma 4.5 *Suppose Z is a background structure, X, Y are background substructures of Z, $U = Atoms(X)$ and $V = Atoms(Y)$.*

1. *If $U \subseteq V$ then the identity on X is the unique embedding of X into Y that is the identity on U.*
2. *If $U \subseteq V$ then $X \leq Y$.*

Proof

1. Suppose that $U \subseteq V$ and let ζ be the identity on U and thus an embedding of U into V. By BC2 (existence), there is an extension of ζ to an embedding η of X into Y and therefore into Z. The identity θ on X is another extension of ζ that is an embedding of X into Z. By BC2 (uniqueness), $\eta = \theta$.

2. follows from 1. □

Lemma 4.6 *In a background structure X, every set U of atoms has an envelope.*

Proof By BC1, there is a background structure Y with $\text{Atoms}(Y) = U$. Let ζ be the identity map on U. By BC2 (existence), ζ extends to an embedding $\eta : Y \to X$. Let Z be the range of η. Clearly, $\text{Atoms}(Z) = U$. By BC0, Z is a background structure and thus a background substructure of X.

By Lemma 4.5, Z is included in every background substructure of X that includes U. This means that $Z = E(U)$. □

It follows that $\{a\}$ has an envelope for every atom a. This is weaker than BC3 which asserts that, in a background structure, every singleton subset has an envelope. Until now we used only BC0–BC2. BC3 does not follow from BC0–BC2, as the following example shows.

Example 4.7 Let K be the class of structures X satisfying the following conditions. The logic elements `true`, `false`, and `undef` are distinct. If $\text{Atoms}(X)$ is empty then X contains no non-logic elements. Otherwise the non-logic part of X consists of atoms and exactly one non-atomic element. It is easy to see that this class K satisfies BC0–BC2. However, if X has more than one atom and x is the unique non-logic non-atomic element, then $\{x\}$ does not have an envelope.

Lemma 4.8 *Every background class has the following property.*

BC3′ *In a background structure X, every $S \subseteq Base(X)$ has an envelope.*

Proof Let $U \rightleftharpoons \bigcup\{\mathrm{Sup}(\{x\}) : x \in S\}$. By Lemma 4.6, U has an envelope $E(U)$. We show that $E(U)$ is also the envelope of S.

$S \subseteq E(U)$. Indeed, for all $x \in S$, $\mathrm{Atoms}(E(\{x\})) = \mathrm{Sup}(\{x\}) \subseteq U$ so that, by Lemma 4.5, $E(\{x\}) \leq E(U)$.

$E(U)$ is the smallest K-substructure of X that includes S. Indeed, let Z be any K-substructure of X that includes S. For every $x \in S$, Z includes $E(\{x\})$ and therefore includes $\mathrm{Sup}(\{x\}) = \mathrm{Atoms}(E(\{x\}))$. Hence Z includes U. Hence Z includes $E(U)$. □

Lemma 4.9 *Every background class has the following property.*

BC3″ *For all $X \in K$, the intersection of any family of K-substructures of X is a K-substructure of X.*

Proof Let F be a family of K-substructures of X. We prove that the substructure $\bigcap F$ is a background structure. Let $U = \bigcap\{\mathrm{Atoms}(Y) : Y \in F\}$. It suffices to prove that $\bigcap F = E(U)$.

$E(U) \leq \bigcap F$. Indeed, by the definition of $E(U)$, $E(U)) \leq Y$ for all $Y \in F$.

$\bigcap F \leq E(U)$. Indeed let x be an element of $\bigcap F$. Every $Y \in F$ contains x, therefore includes $E(\{x\})$, and therefore includes $\mathrm{Atoms}(E(\{x\}))$. It follows that $\mathrm{Atoms}(E(\{x\})) \subseteq U$. By Lemma 4.5, $E(\{x\}) \leq E(U)$ and therefore $E(U)$ contains x. □

This proof gives rise to the following corollary.

Corollary 4.10 *Assume that X is a background structure and let $U_i \subseteq \mathit{Atoms}(X)$ for all $i \in I$. Then*

$$\bigcap_i E(U_i) = E(\bigcap_i U_i)$$

Lemma 4.11 *In Definition 4.1, BC3 can be replaced with BC3″.*

Proof Assume BC3″ and let $X \in K$. Given an element x of X, let F be the collection of K-substructures Y of X such that Y contains x. By BC3″, $\bigcap F$ is a K-substructure of X. Clearly, it is the smallest K-substructure of X that contains x. □

Remark 4.12 The definition of background classes has a simple category-theory version. Consider two categories:

Υ-Str The category of structures for a vocabulary Υ, with embeddings as morphisms.

Set The category of sets, also with embeddings (i.e., one to one maps) as morphisms.

Of course Set is just the special case of Υ-Str, where the vocabulary Υ is empty. But we're interested in the case where Υ contains at least the unary predicate symbol Atomic. This symbol gives rise to a functor $\mathcal{F}$ from Υ-Str to Set. The functor sends each Υ-structure to its set of atoms; on morphisms it acts by restriction. Now a background class is a full subcategory C of Υ-Str that is closed under isomorphisms and under intersections and such that the functor $\mathcal{F}$ when restricted to C is an equivalence of categories C and Set.

Here "full subcategory" means a subclass of the objects, with all of the Υ-Str morphisms between them. And "closed under intersections" should be taken in the category-theoretic sense. In set-theoretic terminology, this means that, given a structure in C and some substructures, also in C, then their intersection should also be in C. One needs the single superstructure to make sense of the intersection.

Lemma 4.13 *Suppose that X is a background structure. For every permutation π of Atoms(X) there is a unique extension of π to an automorphism of X.*

Proof Use BC2 and Lemma 4.4. □

5 Examples of Background Classes

In this section we shall describe some specific background classes. Some of these were the motivation for the general definition of background class.

Recall that the non-obligatory part of a vocabulary is the part that is obtained by removing all logic names as well as the name Atomic.

5.1 Set Background

Up to isomorphism, the non-logic part of a background structure X with atoms U consists of the hereditarily finite sets over U. (See Part 1 about the hereditarily finite sets.) The only non-obligatory basic function of X is the containment relation $\in$. For future reference, this background class will be called the set-background class SB.

There are other versions of set backgrounds. The vocabulary of SB can be enriched in various ways. We consider two ways to do that.

1. The additional basic functions are $\emptyset$, Singleton$(x) = \{x\}$, and

$$\text{BinaryUnion}(x, y) =\rightleftharpoons \begin{cases} x \cup y & \text{if both } x \text{ and } y \text{ are sets} \\ \emptyset & \text{otherwise} \end{cases}$$

The resulting background class is explicitly atom-generated.

2. The additional basic functions are as in [BGS 1999]: $\emptyset$, $\mathrm{Pair}(x, y) \rightleftharpoons \{x, y\}$ and

$$\mathrm{UnaryUnion}(x) \rightleftharpoons \begin{cases} \bigcup_{y \in x} y & \text{if } x \text{ is a set} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathrm{TheUnique}(x) \rightleftharpoons \begin{cases} a & \text{if } x \text{ is a singleton } \{a\} \\ \emptyset & \text{otherwise} \end{cases}$$

Again the resulting background class is explicitly atom-generated.

5.2 String Background

The set of non-logic elements of a background structure with atoms U is the set of strings of elements of U. Let the vocabulary contain symbols for the nullary function Nil (the empty string), the unary function sending a member of U to the one-term string containing it, and the binary operation of concatenation of strings. Then this background class is explicitly atom-generated. If desired, one can introduce additional basic functions $\mathrm{Head}(x)$ and $\mathrm{Tail}(x)$ defined by induction on x. If x is the empty string Nil then $\mathrm{Head}(x) \rightleftharpoons \mathrm{Tail}(x) \rightleftharpoons \mathrm{Nil}$. If $x = ay$ where a is an atom, then $\mathrm{Head}(x) \rightleftharpoons a$ and $\mathrm{Tail}(x) \rightleftharpoons y$.

5.3 List Background

Up to isomorphism, the non-logic part of a backgound structure X with atoms U consist of the lists over U. The terms in the lists can be elements of U or other lists. (So lists differ from the strings in the preceding example in that nesting is allowed.) The non-logic basic functions are Nil and Append. Nil desginates the empty list. Given an atom or a list x and given a list $\langle y_1, \ldots, y_n \rangle$, $\mathrm{Append}(x, y) \rightleftharpoons \langle x, y_1, \ldots, y_n \rangle$. Every list has a unique presentation. As above, this allows us to introduce additional basic functions $\mathrm{Head}(x)$ and $\mathrm{Tail}(x)$ where x ranges over lists. In either version, this background class is explicitly atom-generated.

5.4 Set/List Background

Up to isomorphism, the set of non-logic elements of a background structure with atoms U is the least set V such that

- $U \subseteq V$,
- for every natural number n, if $x_1, \ldots, x_n \in V$ then $\{x_1, \ldots, x_n\} \in V$ and $\langle x_1, \ldots, x_n \rangle \in V$.

Here we do not adopt any of the codings of lists as sets; we regards sets and lists as independent basic constructions.

We leave the choice of the vocabulary to the reader.

5.5 A Non-finitary Background

All example background classes above are finitary. To obtain a non-finitary background class, modify the string-background class by allowing infinite strings.

6 Background Structures and the Reserve

Fix a background class BC. Call the vocabulary Υ_0 of BC the *background vocabulary*, function names in Υ_0 the *background function names*, and members of BC *background structures*.

Definition 6.1 Let A be an algorithm, deterministic or nondeterministic. BC is the *background* of A if the following conditions are satisfied.

• The vocabulary Υ of A includes the background vocabulary Υ_0, and every background function name is static in Υ.

• For every state X of A, the Υ_0-reduct of X (obtained by "forgetting" the basic functions with names in $\Upsilon - \Upsilon_0$) is a background structure. □

Fix a (deterministic or nondeterministic) algorithm A with background BC, and let Υ be the vocabulary of A. The basic functions of A with names in Υ_0 will be called the *background basic functions* of A; the other basic functions of A will be called the *foreground basic functions* of A.

Definition 6.2 Let X be a state of A. An element $x \in \mathrm{Base}(X)$ is *exposed* if x belongs to the range of a foreground function or else x occurs in a tuple that belongs to the domain of a foreground function. □

Recall the property BC3′ of background classes: for every background structure X, every subset of $\mathrm{Base}(X)$ has an envelope in X.

Definition 6.3 The *active part* of a state X of the algorithm A is the envelope of the set of exposed elements; we denote it by $\mathrm{Active}(X)$. The *reserve* of X is the set of atoms of X that do not belong to the active part.

Lemma 6.4 *Every permutation of the reserve of X gives rise to a unique automorphism of X that is the identity on the active part of X.*

Proof Let π be a permutation of the reserve of X. Set $\pi(a) \rightleftharpoons a$ for all atoms a in the active part of X; the extended permutation will be also called π. By Lemma 4.13, there is a unique automorphism θ of the Υ_0-reduct of X that extends π. By definition of active part, any such automorphism that is the identity on the active part is necessarily an automorphism of the full Υ-structure X. □

We remark for future reference that any isomorphism $X \cong Y$ between states of A maps $\mathrm{Active}(X)$ isomorphically onto $\mathrm{Active}(Y)$.

7 Inessential Nondeterminism

The symmetry preservation Lemma 3.1 inspires the following definition.

Definition 7.1 Suppose that A is a nondeterministic algorithm with background BC. A is *essentially deterministic* (or *inessentially nondeterministic*) if the following holds for all states X of A. If (X, X') and (X, X'') belong to τ_A then there is an isomorphism from X' onto X'' that coincides with the identity on Active(X).

Corollary 7.2 *Suppose that A is an inessentially nondeterministic algorithm with background BC. Let $(X, X') \in \tau_A$, $(Y, Y') \in \tau_A$, ζ be an isomorphism from X onto Y, and ζ_0 be the restriction of ζ to Active(X). Then ζ_0 extends to an isomorphism from X' onto Y'.*

Proof By Postulate 4 (Nondeterministic Abstract State), ζ is an isomorphism from X' to some state Y'' with $(Y, Y'') \in \tau_A$. Since A is inessentially nondeterministic, there is an isomorphism $\theta : Y'' \cong Y$ that coincides with the identity on Active(Y), which equals ζ(Active(X)) by the remark at the end of the last section. Then $\theta \circ \zeta$ is an isomorphism from X' to Y' and agrees with ζ on Active(X), i.e., it extends ζ_0. □

Part 3
Nondeterministic Choice Problem for Gandy Machines

8 Gandy Machines

Following Gandy [1980], fix an infinite countable set U of atoms. Recall that HF(U) is the collection of hereditarily finite sets over U (see Part 1). Let $\mathcal{G}$ be the structure

$$(U \cup \mathrm{HF}(U), \in, U)$$

Every permutation π of U naturally extends to an automorphism of $\mathcal{G}$ as follows: if $x \in \mathrm{HF}(U)$ then $\pi x \rightleftharpoons \{\pi y : y \in x\}$. It is easy to see that every automorphism of $\mathcal{G}$ is obtained this way.

A subset S of HF(U) is *structural* if it is closed under automorphisms of $\mathcal{G}$. In other words, S is structural if and only if, for every $x \in S$ and for every permutation π of U, we have $\pi x \in S$. The following definition plays an important role in this part.

Definition 8.1 Let S be a structural subset of $\mathrm{HF}(U)$. A function $F : S \to \mathrm{HF}(U)$ is *structural* if, for every $x \in S$ and for every permutation π of U, there is a permutation ρ of U that pointwise fixes $\mathrm{Sup}(\{\pi x\})$ and such that $\rho\pi Fx = F\pi x$.

Gandy defined only structural functions over $\mathrm{HF}(U)$ but used structural functions over arbitrary structural subsets of $\mathrm{HF}(U)$. The following lemma clarifies the issue of structural functions over $\mathrm{HF}(U)$ vs. structural functions over structural subsets of $\mathrm{HF}(U)$.

Lemma 8.2

1. A structural function F over a structural set S extends to a structural function over HF(U).

2. Suppose that F is a structural function over HF(U) and let S be a structural subset of HF(U). Then the restriction $F|S$ of F to S is a structural function over S.

Proof

1. Set $F(x) \rightleftharpoons \emptyset$ for all $x \in \mathrm{HF}(U) - S$. We show that the extended function F is structural over $\mathrm{HF}(U)$. Let $x \in \mathrm{HF}(U)$ and π be an arbitrary permutation of U. We need to prove the existence of an appropriate permutation ρ. If $x \in S$, then the existence of an appropriate ρ follows from the structurality of F over S. Suppose that $x \notin S$. By the structurality of S, $\pi x \notin S$. Then $\emptyset = Fx = F\pi x = \pi Fx$. The desired ρ is the identity permutation of U.

2. Let $x \in S$ and π be an arbitrary permutation of U. Since F is structural over $\mathrm{HF}(U)$, there is a permutation ρ of U that pointwise fixes $\mathrm{Sup}(\{\pi x\})$ and such that $\rho\pi Fx = F\pi x$. If S contains x then it contains πx as well because S is structural. It follows that $\rho\pi(F|S)x = (F|S)\pi x$ for all $x \in S$. □

Now we are ready to recall the notion of Gandy machine at the level of detail appropriate for this paper.

Definition 8.3 A *Gandy machine* M is a pair (S, F)

- S is a structural subset of $\mathrm{HF}(U)$, and
- F is a structural function from S into S, and
- some additional constraints are satisfied.

Intuitively, S is the set of states of M and F is the one-step transition function. The additional constraints are not important for our purposes in this paper.

9 The Nondeterministic Choice Problem

We start with an example Gandy machine $M^0 = (S^0, F^0)$. S^0 is the collection of finite subsets of U. Obviously, S^0 is structural. If $x \in S^0$, then $F(x) \rightleftharpoons x \cup \{a\}$ where a is an atom in $U - x$. We check that F^0 is structural as well. Suppose that $x = \{a_1, \ldots, a_n\} \in S^0$ where $a_1, \ldots, a_n$ are distinct. Then $F^0 x = \{a_1, \ldots, a_n, b\}$ for some atom $b \notin x$ so that $a_1, \ldots, a_n, b$ are all distinct. Let π be a permutation of U. Then

$$\begin{aligned} \pi x &= \{\pi a_1, \ldots, \pi a_n\} \\ \pi F^0 x &= \{\pi a_1, \ldots, \pi a_n, \pi b\} \quad \text{where } \pi b \notin \pi x \\ F^0 \pi x &= \{\pi a_1, \ldots, \pi a_n, c\} \quad \text{for some } c \notin \pi x \end{aligned}$$

The desired ρ transposes πb and c and leaves other atoms intact.

Thus, M^0 satisfies the part of the definition of Gandy machine that we gave explicitly; the reader familiar with [Gandy 1980] is invited to check that the "additional constraints" that we alluded to are also satisfied.

Now consider an arbitrary Gandy machine $M = (S, F)$ and let $x \in S$. Think about x as the current state of M, so that Fx is the next state of M. It is possible that $\mathrm{Sup}(\{Fx\})$ contains atoms that are not in $\mathrm{Sup}(\{x\})$; that certainly happens in the case of our example machine M^0. The choice of such new atoms should not matter. That is why Gandy requires that F is structural. Two questions arise.

1. Is the structurality of F a correct requirement? That is, does it capture the idea that the choice of new elements is irrelevant?
2. Is there a better solution of this nondeterministic choice problem?

We believe that the answer to the second question is positive. To this end we will propose a nondeterministic formalization of Gandy machines. The answer to the first question is positive as well if one sticks to deterministic machines; we will show that the structurality requirement is equivalent to the intuitively expected requirement that the nondeterministic versions of Gandy machines are essentially deterministic in the sense of Section 7.

10 Nondeterministic (Specifications for) Gandy Machines

Let S be a structural subset of $\mathrm{HF}(U)$ and $F : S \to S$ be any unary operation over S. Think about $M = (S, F)$ as a machine, like a Gandy machine. Of course, we are primarily interested in the case when M is a Gandy machine, but for technical reasons we consider a more general case. We define a nondeterministic algorithm A, or A_M, that may serve as a nondeterministic specification for M.

Definition 10.1 The *nondeterministic specification* of $M = (S, F)$ is the algorithm A defined as follows.

All states of A have the same base set, namely the set $U \cup \mathrm{HF}(U)$ extended with three additional elements (interpreting) $\mathtt{true}, \mathtt{false}, \mathtt{undef}$. A has only three non-logic basic functions. Two of them are static: the unary relation Atomic and the containment relation $x \in y$. The third basic function is a nullary dynamic function Core that gives the current state of M. For brevity, let $\mathrm{Core}(X)$ be the value of Core at state X and let $\mathrm{Sup}(X) \rightleftharpoons \mathrm{Sup}(\{\mathrm{Core}(X)\})$.

The one-step transition relation τ_A consists of pairs (X, Y) of states of A such that $\mathrm{Core}(Y) = F(\mathrm{Core}(X))$ or, more generally, there is a permutation π of U that pointwise fixes $\mathrm{Sup}(X)$ and such that $\mathrm{Core}(Y) = \pi F(\mathrm{Core}(X)))$.

To explain this definition, consider the example M^0 from the preceding section. Abbreviate A_{M^0} to A^0. Abbreviate τ_{A^0} to τ^0. In this situation, $\mathrm{Sup}(X) = \mathrm{Core}(X)$ for all states X of S. Fix a particular state X of A^0. What are the states Y such that $(X, Y) \in \tau^0$? It is easy to see that these are exactly the state Y such that $\mathrm{Sup}(Y)$ consists of the atoms in $\mathrm{Sup}(X)$ plus one additional atom a. Any atom in $U - \mathrm{Sup}(X)$ will do. No atom in $U - \mathrm{Sup}(X)$ has preferential treatment or is discriminated against.

Remark 10.2 Gandy does not specify initial states of his machines. Accordingly we ignore initial states as well.

It is easy to see that A is a nondeterministic algorithm with background SB described in Subsection 5.1. The only exposed element of a state X of A is $\mathrm{Core}(X)$. Accordingly the active part $\mathrm{Active}(X)$ of X is $\mathrm{Sup}(X) \cup \mathrm{HF}(\mathrm{Sup}(X))$ plus the elements $\mathtt{true}, \mathtt{false}, \mathtt{undef}$. Hence $\mathrm{Reserve}(X) = U - \mathrm{Sup}(X)$.

We saw that every permutation π of U extends to an automorphism of the structure $\mathcal{G} = (U \cup \mathrm{HF}(U), \in, U)$. However, π is not necessarily an automorphism of X because it can move $\mathrm{Core}(X)$. It is an automorphism of X if and only if $\pi(\mathrm{Core}(X)) = \mathrm{Core}(X)$. Identify a permutation π of $\mathrm{Reserve}(X)$ with the permutation of U which pointwise fixes $\mathrm{Sup}(X)$ and coincides with π on $\mathrm{Reserve}(X)$.

Corollary 10.3 *Let (X, Y) be states of A. Then $(X, Y) \in \tau_A$ if and only if there is an permutation π of $\mathrm{Reserve}(X)$ such that $\mathrm{Core}(Y) = \pi F(\mathrm{Core}(X))$.*

11 Essential Determinism and Structurality

Let S be a structural subset of $\mathrm{HF}(U)$ and F be any unary operation over S. Further let A be the nondeterministic specification for (S, F). Abbreviate τ_A to τ.

Theorem 11.1 *The following are equivalent.*

1. *A is essentially deterministic.*
2. *F is structural over S.*

Proof First we assume 1 and prove 2. Let $x \in S$ and let π be any permutation of U. We construct the desired ρ.

Let X be the state of A with $\mathrm{Core}(X) = x$ and let πX be the state of A with $\mathrm{Core}(\pi X) = \pi x$. Since S is structural, it contains πx and thus πX is a legitimate state of A.

View π as an isomorphism from X to πX. Since A is essentially deterministic, there is an isomorphism η from $\tau(X)$ to $\tau(\pi X)$ which coincides with π on $\mathrm{Active}(X)$ and in particular on $\mathrm{Sup}(\{x\})$. The desired $\rho z \rightleftharpoons \eta\pi^{-1}z$ for all $z \in \mathrm{HF}(U)$.

If $\pi a \in \mathrm{Sup}(\{\pi x\})$ then $\rho(\pi a) = \eta\pi^{-1}\pi a = \eta a = \pi a$. Thus ρ is the identity on $\mathrm{Sup}(\{\pi x\})$.

Since η is an isomorphism from τX onto $\tau(\pi X)$, we have $\eta\mathrm{Core}(\tau X) = \mathrm{Core}(\tau\pi X)$, that is $\eta Fx = F\pi x$. Therefore $\rho\pi Fx = \eta\pi^{-1}\pi Fx = \eta Fx = F\pi x$.

Second we assume 2 and prove 1. Suppose that (X, Y) and (X, Z) belong to τ. We need to prove that there is an isomorphism from Y onto Z that pointwise fixes $\mathrm{Active}(X)$.

Without loss of generality $\mathrm{Core}(Y) = F(\mathrm{Core}(X))$. Indeed, suppose that 2 is proved in this special case. Now consider the general case, and let X' be the state of A with $\mathrm{Core}(X') = F(\mathrm{Core}(X))$. By the special case of 2, there is an isomorphism ζ from X' onto Y that pointwise fixes $\mathrm{Active}(X)$. Similarly, there is an isomorphism η from X' onto Z that pointwise fixes $\mathrm{Active}(X)$. Then $\eta \circ \zeta^{-1}$ is an automorphism from Y onto Z that pointwise fixes $\mathrm{Active}(X)$.

Let $x \rightleftharpoons \mathrm{Core}(X), y \rightleftharpoons \mathrm{Core}(Y)$ and $z \rightleftharpoons \mathrm{Core}(Z)$. We have $y = Fx$. Since $(X, Z) \in \tau$, there exists a permutation π of U that pointwise fixes $\mathrm{Sup}(\{x\})$ and such that $z = \pi Fx$. Since F is structural, there exists a permutation ρ of U that pointwise fixes $\mathrm{Sup}(\{\pi x\})$ and such that $\rho\pi Fx = F\pi x$. Since π pointwise fixes $\mathrm{Sup}(\{x\})$, we have $\pi x = x$ so that ρ pointwise fixes $\mathrm{Sup}(\{x\})$ (and therefore pointwise fixes $\mathrm{Active}(X)$) and $\rho z = \rho\pi Fx = F\pi x = Fx = y$. Then ρ^{-1} pointwise fixes $\mathrm{Active}(X)$ and takes y to z. □

References

BGS 1999 Andreas Blass, Yuri Gurevich and Saharon Shelah, "Choiceless Polynomial Time", Annals of Pure and Applied Logic 100 (1999), 141–187.

Enderton 1972 Herbert B. Enderton, "A Mathematical Introduction to Logic," Academic Press.

Gandy 1980 Robin Gandy, "Church's thesis and principles for mechanisms" in: "The Kleene Symposium" (ed. J. Barwise et al.), North-Holland, 1980, 123–148.

Gurevich 1991 "Evolving Algebras: An Attempt to Discover Semantics", Bulletin of European Assoc. for Theor. Computer Science, no. 43, Feb. 1991, 264–284. A slightly revised version appeared in G. Rozenberg and A. Salomaa, editors, "Current Trends in Theoretical Computer Science", World Scientific, 1993, 266–292.

Gurevich 1995 Yuri Gurevich, "Evolving Algebra 1993: Lipari Guide", in "Specification and Validation Methods", Ed. E. Boerger, Oxford University Press, 1995, 9–36.

Gurevich 2000 Yuri Gurevich, "Sequential Abstract State Machines Capture Sequential Algorithms", ACM Transactions on Computational Logic 1, to appear.