THE EVOLUTION OF ALGORITHMIC LEARNING RULES: A Global Stability Result*

Luca Anderlini
(St. John's College, Cambridge)

Hamid Sabourian
(King's College, Cambridge)

October, 1995

ABSTRACT

This paper considers the dynamic evolution of algorithmic (recursive) learning rules in a normal form game. It is shown that the system — the population frequencies — is *globally stable* for any arbitrary N-player normal form game, if the evolutionary process is algorithmic and the 'birth process' guarantees that an appropriate set of 'smart' rules is present in the population. The result is independent of the nature of the evolutionary process; in particular it does not require in any way the dynamics of the system to be 'monotonic in payoffs' — those rules which do better in terms of payoffs grow faster than those who do less well. The paper also demonstrates that any limit point of the distribution of actions in such an evolutionary process corresponds to a Nash equilibrium (pure or mixed) of the underlying game if the dynamics of the system are continuous and monotonic in payoffs.

JEL Classification: C70, C72, C79, D83.

Keywords: Evolution, Learning Rules, Computability, Monotonic Dynamics.

Proposed Running Head: Algorithmic Learning Rules.

Address for correspondence: Luca Anderlini, Faculty of Economics & Politics, Sidgwick Avenue, Cambridge CB3 9DD, U.K. (e-mail la13@econ.cam.ac.uk).

* We are grateful to seminar participants at U.C.L.A., L.S.E., Stanford, Yale, Minnesota, Columbia, Essex, Harvard and Tel-Aviv University for stimulating comments. Any remaining errors are our own responsibility.

1. Introduction

1.1. Motivation

Broadly speaking, in game theory there are two approaches to the justification of solution concepts such as Nash equilibrium. First, there is the classical game (decision) theory approach. Ken Binmore (Binmore 1987, Binmore 1988) calls this the 'eductive' approach. The second is the learning/evolutionary approach — 'evolutive' to use Binmore's terminology again.

In the 'eductive' approach, agents are rational (Bayesian) optimizers. There is common knowledge of the structure of the game, and the beliefs of the agents are consistent (or even common knowledge). Agents work out 'the solution' to the game and play accordingly. There are no mistakes and 'the solution' is achieved through careful reasoning by the agents. In a way, the agents are as 'complex' as the environment in which they play and thus they can 'solve' the game (Aumann and Brandenburger 1995).

Clearly, this approach attributes too much in terms of computational ability, information and knowledge to the agents. Moreover, it does not, on its own, offer an explanation as to why agents' beliefs should be consistent or common knowledge. Finally, it has become commonly accepted that this approach on its own cannot be used convincingly to select among the multitude of equilibria which often occur in game-theoretic models.

In the second approach, the agents do not necessarily have the ability, information and/or adequate beliefs to 'solve' the game. The system evolves by agents taking decisions, making mistakes, learning, imitating, revising their early decisions and so on. An equilibrium in this approach corresponds to a steady state of a dynamical process. Selection among the set of equilibria can sometimes be achieved considering the stability (in some sense) of the steady states of the system.

Not only the evolutionary models provide a natural setting for modelling bounded rationality, but they are often said to be necessarily models of bounded rationality. This is because in these models players, by assumption, are less sophisticated than the environment they face. They make mistakes, learn and imitate. They do not

(and cannot) 'solve' the game prior to playing it. The environment is too 'complex' for them.

This paper belongs to the evolutive approach. Our objective is to show that under general assumptions, deterministic evolutionary models with learning rules are globally stable if a sufficiently rich class of learning rules are present in the system. Although we do not introduce complexity considerations, an element of bounded rationality is explicitly present in the model we develop: we only consider learning rules which are algorithmic. Formally, these are rules which can be computed [1] by Turing machines.

1.2. Standard Evolutionary Models

The 'standard'[2] evolutionary model consists of $N \geq 2$ large (often infinite) populations of myopic and unsophisticated players (types) playing some underlying game repeatedly. At each period, every type x belonging to some population i is randomly matched with some type of all other populations to play an underlying N-player normal form game $\Gamma = \{\mathcal{A}_i, \pi_i\}_{i=1}^{N}$ where $\mathcal{A}_i$ is the set of actions available to types belonging to population i and π_i represents their payoffs at each period.

The 'selection dynamics' of these models determine how the proportion of each type in each of the N populations changes as a function of how well that type has done in the previous period in terms of payoff and the distribution of payoffs of all other types.[3] The interpretation being that each type in population i represents (plays) an action $a_i \in \mathcal{A}_i$ and those actions/types who do well in terms of payoff grow faster. In the biological literature, payoffs represent fitness and therefore it is natural to assume that the proportions of types adjust in response to payoff differences. In a non-biological setting, the selection dynamics represent learning and

[1] Throughout the paper we use the terms (partial) recursive, algorithmic and computable in an interchangeable way. See Section 2 below for a general discussion of the computability framework.

[2] See the excellent surveys in VanDamme (1987), Hofbauer and Sigmund (1988) and Weibull (1992) (which also contain many original results). Marimon and McGrattan (1992) provide a survey of the related literature on learning dynamics.

[3] Throughout the paper, we concentrate on *deterministic* selection dynamics, in the sense that we take the growth rate of each type in each period to be deterministic. There is also a growing literature on stochastic selection dynamics, which has yielded very promising results, particularly concerning the problem of selection among different Nash equilibria. Among others, we recall the work of Foster and Young (1991), Kandori, Mailath, and Rob (1993) and Young (1993).

imitation from past experience. Ideally, one would like to build up the selection dynamics from an explicit theory of how individual players switch between different actions/types. There is a growing literature which analyses formal models of learning and imitation behaviour based on the information available to the players (Brown 1951, Canning 1992a, Fudenberg and Kreps 1991, Fudenberg and Levine 1993a, Fudenberg and Levine 1993b, Jordan 1991, Kalai and Lehrer 1993, Milgrom and Roberts 1990, Milgrom and Roberts 1991, Selten 1991, to name just a few).

In many cases, the evolutionary approach has tried to avoid constructing a (possibly arbitrary) theory of learning and imitation by placing assumptions directly on the selection dynamics and hoping that these properties are general enough to include processes produced by a variety of learning and imitation theories. A general class of selection dynamics which has been extensively studied, is the class of *monotonic dynamics*. The chief feature of this class of selection dynamics is that at any time those types which do better in terms of payoff grow faster than those which do worse. A special case of monotonic dynamics (common in the biological literature) are the *replicator dynamics* (cf. Definition 14 below). In this case, the growth rate of each type within a population is simply the difference (normalized) between that type's payoff and the population's average payoff.

Starting with the early example of Shapley (1964) on fictitious play, results on stability and convergence of evolutionary and learning models with no noise are not encouraging. Positive results on global convergence exist only for a very specific class of games, namely for 'dominant solvable' games (Nachbar 1990, Milgrom and Roberts 1991).

In the next subsection we give an informal explanation for the stark difference between the results of this paper — global stability with a general N-player underlying game — and the findings of the previous literature.

1.3. Outline of Model and Results

In the standard evolutionary model, a type within population i is identified with an action $a_i \in \mathcal{A}_i$ in the underlying game [4]; a given type takes the same action at

[4]See footnote 2 above. A notable exception to the standard evolutionary model is the work of

every period. A key difference between the model we develop in this paper and the previous literature lies in the definition of type: in this paper we identify a type with an algorithmic learning rule. In our model, a type in population i is a computable map from some information set related to (but not necessarily coinciding with) the history of the system so far, into the set of actions $\mathcal{A}_i$. In the model we develop, the same type can take different actions at different times.

Our second departure from the existing literature is really a generalization of the standard evolutionary model. It concerns the way 'new types' may appear into the system at different times, and some types may become extinct in finite time. The dynamics of our model are given by three distinct elements: a standard (except for a computability assumption) selection dynamics, a *birth* process and a *death* process. The three are modeled as separate parts purely for analytical convenience. The model starts with a finite set of types in each population. The selection dynamics component of the model does not provide for the entry of new types or the extinction of any existing types in finite time: it only defines what the growth rate of each type would be in the absence of birth and death. The (computable) birth and the death processes determine which (finite) sets of types appear and disappear from the population at each time t. The birth process enables us eventually to bring into the evolutionary system a 'sufficiently rich' set of types. This is crucial to our stability results as we will outline shortly.[5] The death process is actually redundant from the point of view of our results, and very few assumptions are made on it: it is enough that the 'top performers' at t in terms of the selection dynamics do not go extinct between t and $t+1$. We introduce it purely for reasons of symmetry with the birth process. We refer to the three components of the dynamics (the selection dynamics, the birth process, and the death process) taken together as the *overall dynamics* of the model.

The third difference between our model and the standard evolutionary model lies in our assumptions of computability. In essence, we assume that the entire model is computable within Church's thesis. This requires computability of the initial distribution of types in each population, of the selection dynamics, of the birth and death

Blume and Easley (1992). Their definition of types is similar to the one we propose here.

[5]In Section 6.1, we outline how our results may hold with the birth and death processes replaced by an assumption of 'large (infinite) support' on the initial distribution of types.

processes and of the map between information and actions which defines each type.

This paper contains two sets of results. Very informally, the first set of results say that the proportion of each type x in population i at time t, denoted by $P_i^t(x)$, converges if the birth process is guaranteed eventually to bring into each population a 'sufficiently rich' set of types. We present two, related, sets of conditions which guarantee that the sufficiently rich 'support' condition we have just mentioned is satisfied. The second result says that if, in addition to the assumptions needed for stability, the selection dynamics are monotonic (those with higher expected payoff are rewarded with higher growth rates) and continuous, then every limit point of the distribution of actions (obtained from the distribution of types) corresponds to a Nash equilibrium (possibly mixed) of the underlying game.

Our stability results do not depend on any assumption on the shape of the selection dynamics, except that they should be algorithmic. They are even consistent with selection dynamics which reward those types which do badly in terms of payoff.

The second result is not so surprising and is very similar to those of Nachbar (1990) and Samuelson and Zhang (1992) among others. It is important to notice that although probabilities over types converge (the first result), probabilities over actions do not necessarily converge. This is because types can condition their actions on time and thus can take different actions at different times. Therefore, even when the first result holds, in the limit some (or all) types could be switching between different Nash equilibria of the underlying game. An immediate corollary of the second result is that if the underlying game has a unique Nash equilibrium then probabilities over actions converge as well (to the unique Nash equilibrium).

The key to the argument which underlies our first result (stability) can be intuitively outlined as follows. We start by establishing that for any profile of initial distributions and overall dynamics, and for any population i, there exists a type $\overline{x}_i$, referred to as the 'smart' type (or smart machine), which can grow as fast as any other type in population i at each time t. The existence of the smart machine is proved by constructing another program which, upon receiving the initial distribution as an input, simulates the entire evolutionary system and computes an action which maximizes the growth rate in each period. The smart machine produces the same output as this program in each period. Thus, in a sense, the smart machine

behaves 'as if' it simulated the dynamics in order to take an action which maximizes its growth rate at each time t.

Suppose now that the smart type, which takes a sequence of actions that ensures growth at the maximal rate for all t, either is in the support of the initial distribution or appears into the system at some time through the birth process. Then the system converges because the probability of the smart type is monotonically increasing and thus it can be used to play a role very similar to a Lyapunov function for the overall dynamics. The first step in the proof of our stability results is in demonstrating the existence of smart types such as the ones we have just described informally. The argument is then closed making assumptions which ensure that the birth process (or the initial distribution) is such that the relevant smart types eventually appear into the system. We present two, related, sets of conditions which ensure that this is the case. We start with a 'grain of truth' condition (which is reminiscent of Kalai and Lehrer (1993)) which simply states that *the* smart machine which behaves 'as if' it simulated the actual dynamics must appear in the system at some point in time.

Our second stability result relies on a 'regularity' assumption on the set of Turing machines which define the overall dynamics of the system. Corresponding to the set of overall dynamics which obey this restriction, we are able to identify a well behaved set of 'potential smart types'. We then obtain stability, provided that the overall dynamics obey the regularity restriction, and provided that the entire set of potential smart types are eventually introduced into the system. We then go on to show that in order to ensure that the overall dynamics satisfy the regularity condition needed, it is enough to restrict attention to those dynamical systems which satisfy a 'provability' condition which seems intuitively weak and hence particularly appealing. Very informally, the provability condition we impose amounts to restricting attention to those dynamics which can be computed by a set of Turing machines *and* such that a 'proof' exists that the given set of Turing machines compute the given dynamics.

To establish our stability results, we have departed from the standard evolutionary theory in two main ways: (i) we have allowed for types which can condition their actions at least on time and (ii) we have assumed the entire model (the types, the overall dynamics and the initial distribution of types) to be algorithmic. (We shall define the precise meaning of this statement shortly.)

Are (i) and (ii) necessary for establishing our stability results? We think (i) is necessary because the smart type may need to take different actions at different times to ensure that it grows faster than others at different periods. Restricting the model to be algorithmic (ii) plays a three-fold role in the analysis below. Firstly, it makes the set of all possible types a countable one. As a result, it is possible, at least in principle, that all possible smart types are eventually given strictly positive weight in the dynamics.[6] Secondly, by appealing to the existence of the universal program, which can simulate all other programs, we can show that there exists a program which can simulate the entire model. Thirdly, we are able to appeal to a particular 'parameterization' theorem which shows that we can find an algorithmic learning rule which behaves 'as if' it simulates the entire model like the universal program. We return to these issues in the concluding section of the paper.

The outline of the paper is as follows. In Section 2 we introduce the recursive function framework which we use in the rest of the paper. In Section 3 we present the model in full detail. Section 4 contains our stability results. In Section 5 we show that, under some additional conditions, the limit points of the action frequencies in our model correspond to the Nash equilibria of the underlying game. Section 6 concludes the paper discussing one extension and one possible alternative formulation of our model, as well as offering some further remarks on the role which the computability framework plays in our analysis.

The paper has four appendices. In Appendix A we present all the basic results from the computability literature which are used in the paper. For ease of exposition, most of the proofs of our results are not in the text, but can instead be found in Appendix B. Appendix C contains all the material concerning the 'provability' restriction which we use in our second stability result. Finally Appendix D outlines how our results can be extended to the case of computable functions defined over 'computable reals'. A prefix of 'A', 'B', 'C' or 'D' in the numbering of a definition, theorem, equation etc. indicates that the relevant item is to be found in the corresponding appendix.

[6] With a continuum of types, it may be impossible to attach an 'atom' of probability to *all* potential smart types, since this set may also have the cardinality of the continuum.

2. Algorithms, Computability and Turing Machines

Intuitively, an algorithm is a clerical (mechanical, deterministic) procedure which can be applied to a string of symbols as an input and which eventually yields a corresponding string of symbols as an output. Moreover, input and output come from a fixed alphabet, and the procedure is given as a set of instruction of finite size.

Turing machines are a means of formalizing of the above intuitive notion. There have been many different formal characterizations of the notion of algorithm, but all have turned out to be equivalent (Cutland 1980, Ch. 3). Turing machines (or their equivalent) represent what is widely regarded in mathematics as the appropriate notion of effective computability in the widest possible sense. They represent a 'most powerful' class of computing devices — anything that a mathematician can compute can also be computed by a Turing machine (this claim is also known as Church's thesis).

The idea that players in a game can be thought of as a computing devices is not new (Aumann 1981, Neyman 1985, Rubinstein 1986, Abreu and Rubinstein 1988, to name a few). Most of the available literature concentrates on a class of computing devices known as finite automata, and on the Nash equilibria of the 'machine game'. Following Binmore (1987) and Anderlini (1989), we take the view that Turing machines (and thus general computability) are the appropriate class of programs for modelling many game-theoretic situations.[7]

A Turing machine is essentially identified by its 'program'. A program is a finite string of symbols from a fixed alphabet which obey some syntactical rules. It follows that, using a standard technique known as Gödel numbering (see Theorem A.1) the set of all Turing machines can be put in a one-to-one (computable) correspondence with the set of natural numbers $\mathbb{N}$. The machines' inputs and outputs are also finite strings of symbols from a fixed alphabet. It follows that the same technique can be used to code and decode the machines' inputs and outputs into the naturals. The detailed specification of Turing machines is irrelevant to the analysis which follows.[8] Following

[7] McAfee (1984), Megiddo (1986), Megiddo and Widgerson (1987), Howard (1988), Spear (1989), Anderlini (1990), Canning (1992b), and Anderlini and Sabourian (1995) also model players as Turing machines.

[8] All the relevant results on algorithmic functions can be found in Appendix A.

standard notation in the computability literature $\varphi_x(y)$ will describe the result of the computation of Turing machine with Gödel number $x \in \mathbb{N}$ when applied to the input string coded by the Gödel number $y \in \mathbb{N}$. The notation $\varphi_x(y) \uparrow$ indicates that the computation $\varphi_x(y)$ does not halt (it loops), while the notation $\varphi_x(y) \downarrow$ denotes the fact that the computation $\varphi_x(y)$ does halt.

DEFINITION 1 *(Computable Function): A partial function C from $\mathbb{N}^m$ to $\mathbb{N}$ is said to be computable if and only if $\exists c \in \mathbb{N}$ such that $C(y_1, \ldots y_m) \simeq \varphi_c(y_1, \ldots, y_m)$ $\forall (y_1, \ldots, y_m) \in \mathbb{N}^m$, where the symbol $\simeq$ stands for 'defined on the same set and equal whenever defined'. A computable function which is defined for every possible input is called a 'total' computable function. Throughout the paper, we adopt the notational convention of denoting computable functions by capital letters. A Turing machine which computes a given function will be denoted by the same letter in lower case; thus machine c computes function C.*

3. The Model

3.1. The Underlying Game, Histories and Types

The N-player normal form game underlying our model is denoted by $\Gamma = \{\mathcal{A}_i, \pi_i\}_{i=1}^N$ where $\mathcal{A}_i$ is the set of actions available to player i, $\mathcal{A}$ denotes $\bigtimes_{i=1}^N \mathcal{A}_i$, $\pi_i : \mathcal{A} \to \mathbb{R}$ represents i's payoffs, and $\pi : \mathcal{A} \to \mathbb{R}^N$ represents the payoff vectors. Lastly, A_i and A denote the cardinalities of $\mathcal{A}_i$ and $\mathcal{A}$ respectively.

Time is discrete and is indexed by $t = 0, 1, \ldots$ The evolutionary system consists of N populations, one for each player i of Γ. At every time period t, each population consists of a finite number of types to be defined shortly, although the set of different types in each population can vary through time. Let the set of types present in population i at time t be denoted by $\mathcal{T}_i^t$, with cardinality N_i^t; we also let $\mathcal{T}^t = \{\mathcal{T}_1^t, \ldots, \mathcal{T}_N^t\}$, and $N^t = \sum_{i=1}^N N_i^t$.

Given a type $x \in \mathcal{T}_i^t$, we denote by $a_i^t(x) \in \mathcal{A}_i$ the action which type x takes at time t. We also define $a_i^t \equiv \{a_i^t(x)\}_{x \in \mathcal{T}_i^t}$ to be the action profile of all types in population i at time t. Finally, we let $a^t \equiv \{a_1^t, \ldots, a_N^t\}$ be the action profile of all types in all populations at time t.

The actual sets of types present in each population at t will depend on the birth and death processes to be defined in Section 3.3 below. However, given a sequence of type sets $\{\mathcal{T}^0, \mathcal{T}^1, \ldots\}$, we can already define the outcome path of the system up to t. An *outcome path* of the system of length t is denoted by $\hat{h}^t \equiv \{a^0, \ldots, a^{t-1}\}$. By convention, we set $\hat{h}^0 = \emptyset$. The set of all possible outcome paths of length t is denoted by $\hat{\mathcal{H}}^t$, while the set of all possible finite outcome paths (of any length) is denoted by $\hat{\mathcal{H}}$. Notice that since the type sets $\mathcal{T}^t$ are all finite, the set of all possible finite outcome paths can be coded into the natural numbers in a standard way (see Theorem A.1). With a slight abuse of notation, throughout the paper we will use the same symbol for any outcome path $\hat{h}^t$ and its code in $\mathbb{N}$; thus $\hat{\mathcal{H}} \subset \mathbb{N}$.

We are now ready to define the information about $\hat{h}^t$ which is given to the types at each period. We call this the information (about $\hat{h}^t$) which types receive at t, and we denote it by h^t. The values of h^t define a *partition* of the set of all possible finite outcome paths, $\hat{\mathcal{H}}$. Therefore, for a *given* partition, the set of all possible values of h^t, denoted by $\mathcal{H}$, can also be viewed as a subset of the natural numbers $\mathbb{N}$. As for the outcome paths, with a slight abuse of notation we use the same symbol for any element of $\mathcal{H}$ and its code in $\mathbb{N}$.

Definition 2 *(Information): The information which types receive at time t is determined by some function Z from $\hat{\mathcal{H}}$ to $\mathcal{H}$. In other words, at each t, each type receives as input a natural number $h^t = Z(\hat{h}^t)$. The function Z is called the information function of the model.*

Notice that, for the time being, we are not imposing any assumptions on the information function Z. Indeed, Definition 2 above is consistent with both the possibility that h^t coincides with $\hat{h}^t$ and that of h^t containing no information at all about $\hat{h}^t$. We restrict the form of Z in Section 3.4 below (Assumptions 4 and 5).

Types are algorithms (Turing machines) which map the information which is given to them at date t into an action at t. It is therefore natural to restrict attention to Turing machines which, given any input, halt and yield an action in $\mathcal{A}_i$.

Definition 3 *(Allowable Algorithms): A Turing machine $x \in \mathbb{N}$ is an 'allowable'*

machine (or type) for population i if and only if

$$\varphi_x(n) \downarrow \in \mathcal{A}_i \quad \forall\, n \in \mathbb{N}$$

The set of allowable machines for population i is denoted by $\mathcal{M}_i$ throughout the paper.[9]

We are now ready to give a formal definition of types in our model.

DEFINITION 4 *(Types): A type for population i is an allowable Turing machine for population i; thus $\mathcal{T}_i^t \subset \mathcal{M}_i \subset \mathbb{N}$ for all $t = 0, 1 \ldots$ and for all $i = 1, \ldots, N$. The action of type $x \in \mathcal{T}_i^t$ at t is the output of Turing machine x on input h^t; therefore $a_i^t(x) \equiv \varphi_x(h^t)$.*

Notice that given a sequence of vectors of type sets $\{\mathcal{T}^0, \mathcal{T}^1, \ldots\}$, and an information function Z, the outcome path of the system can be generated in a forward recursive way, similar to the procedure which generates the outcome path of a repeated game. The outcome path of length 1, $\hat{h}^1$ can be obtained by carrying out all the computations $\varphi_x(h^0)$ for all $x \in \mathcal{T}_i^0$ and for all i. Given $\hat{h}^1$, the information function Z, yields h^1. At this point the computations $\varphi_x(h^1)$ can be carried out, for all $x \in \mathcal{T}_i^1$ and for all i, in order to obtain $\hat{h}^2$ and so on, forward through time.

Before we proceed any further, it is helpful to emphasize the following two points about the information function and types.

REMARK 1: *The types we have defined can condition their actions on h^t. In the standard evolutionary model, types are identified with a fixed action through time. In the model we have developed here this could be the case if either the information function Z is a constant function, or if the allowable Turing machines in each population, i, were restricted to machines which yield a constant output in $\mathcal{A}_i$.*

[9] Notice that Definition 3 above implies that we are excluding from the set of possible types any non-halting Turing machines (cf. Section 2 above). This keeps matters simple but is by no means essential to our results. In previous versions of the paper we allowed for non-halting types and obtained similar results. This can be achieved assuming that non-halting types are not rewarded by the selection dynamics so that their extinction is guaranteed in the limit. In a different context, the analysis in Anderlini and Sabourian (1995) allows for non-halting Turing machines.

REMARK 2: *We are implicitly assuming that information is symmetric in our model. This is because we have taken the information function Z to be the same for all types. This is not necessary for our results. We discuss how this assumption can be relaxed in Section 6.1 below.*

3.2. The Selection Dynamics

As we mentioned in Section 1.3, the overall dynamics of our model are made up of three components: the selection dynamics, the birth process and the death process. We introduce these separately, starting with the selection dynamics, purely for ease of exposition. First of all we need some further notation.

We denote by $P_i^t(x)$ the probability (or frequency) of type $x \in \mathcal{T}_i^t$ in population i at time t. Throughout the paper the symbol Δ^k denotes the unit simplex in $\mathbb{R}^k$, and $\text{int}\Delta^k$ denotes its interior. The probability distribution over population i at time t is denoted by $P_i^t \equiv \{P_i^t(x)\}_{x \in \mathcal{T}_i^t} \in \text{int}\Delta^{N_i^t}$. The probability distribution profile over all populations at time t is denoted by $P^t \equiv \{P_1^t, \ldots, P_N^t\} \in \text{int}\Delta^{N^t}$.

We will assume that the growth rate of each type at time t depends in a rather general way on that type's action and on the *distribution* of actions in all populations at time t. Let $\mathcal{W}^t(a_i) \equiv \{x \in \mathcal{T}_i^t \mid \varphi_x(h^t) = a_i \in \mathcal{A}_i\}$. The probability (or frequency) of action a_i in population i at time t is then simply given by $Q^t(a_i) \equiv \sum_{x \in \mathcal{W}^t(a_i)} P_i^t(x)$. It is now possible to define the probability distribution over actions in population i at time t as $Q_i^t \equiv \{Q^t(a_i)\}_{a_i \in \mathcal{A}_i} \in \Delta^{A_i}$. Finally the profile of probability distributions over actions at time t is defined as $Q^t \equiv \{Q_1^t, \ldots, Q_n^t\} \in \Delta^A$.

Excluding the effect of birth and death at $t+1$, the growth rate of an existing type, x, in population i between time t and $t+1$ is denoted by $g_i^t(x)$. The general form of the selection dynamics which we use below is the following, for all i

$$g_i^t(x) = G_i\left[a_i^t(x), Q^t\right] \quad \forall\, x \in \mathcal{T}_i^t \;\; \forall\, t = 0, 1, \ldots \tag{1}$$

with the restriction that

$$g_i^t(x) > -1 \;\text{ and }\; \sum_{x \in \mathcal{T}_i^t} P_i^t(x) g_i^t(x) = 0 \quad \forall\, i = 1, \ldots, N \quad \forall\, t = 0, 1, \ldots \tag{2}$$

so that the dynamics 'remain in the simplex' at all times.

We denote by G the vector of functions $\{G_1, \ldots, G_N\}$ for the rest of the paper. Note that equation (1), together with Definitions 2 and 4, implies that $g_i^t(x)$ depends on the distribution over types P^t (via Q^t), on the value of h^t (since $a_i^t(x) = \varphi_x(h^t)$), and therefore on the outcome path $\hat{h}^t$ and on the information function Z. Before proceeding any further, it is worth emphasizing one key feature of the selection dynamics implicit in (1).

REMARK 3 *(Anonymous Selection Dynamics): The selection dynamics defined by (1) and (2) are anonymous in the sense that the growth rate at t of a particular type x does not depend directly on the type's identity; only the action which x takes at time t matters in terms of its growth rate. This is crucial for the possibility of constructing the 'smart types' mentioned in Section 1.3, since otherwise the selection dynamics could be constructed to 'punish' (in terms of growth rate) the potential smart types on the basis of their identity, whatever the action taken.*

3.3. Birth, Death and the Overall Dynamics

As we discussed in Section 1.3, the possibility of new types entering the system through time is crucial to our results. This is because our stability results rests on the appearance of a sufficiently rich set of types in each population. We model both birth and death at time t as functions of the outcome path of the system of length t.

The set of newborn types at t in population i is characterized by a finite subset $\mathcal{B}_i^t$ of $\mathcal{M}_i$ (cf. Definition 3) and associated positive numbers which represent the frequencies (or probabilities) of each new type in $\mathcal{B}_i^t$.

DEFINITION 5 *(Birth Function): A birth function for population i is a map B_i which for each outcome path $\hat{h}^t$ yields a (possibly empty) finite subset $\mathcal{B}_i^t$ of $\mathcal{M}_i$, and an array of positive frequencies $\{P_i^t(x)\}_{x \in \mathcal{B}_i^t}$, one for each element of $\mathcal{B}_i^t$, satisfying $\sum_{x \in \mathcal{B}_i^t} P_i^t(x) < 1$. By convention, the map B_i on input $\hat{h}^0$ yields the 'initial pair' $\{\mathcal{T}_i^0, P_i^0\}$, with $\mathcal{B}_i^0 = \mathcal{T}_i^0 \neq \emptyset$ and $\sum_{x \in \mathcal{B}_i^0} P_i^0(x) = 1$. Moreover, for analytical convenience, a birth function is assumed to be such that $\mathcal{B}_i^t \cap \mathcal{T}_i^{t-1} = \emptyset$. In other words, a birth function is precluded from introducing into the dynamics any type which is already present in*

the system. The vector of birth functions $\{B_1, \ldots, B_N\}$ *is denoted by* B *for the rest of the paper.*

The death function of the model simply maps outcome paths into subsets of existing types, which then disappear from the system.

DEFINITION 6 *(Death Function): A death function for population* i *is a map* D_i *which for each outcome path* $\hat{h}^t$ *yields a (possibly empty) subset,* $\mathcal{D}_i^t$*, of* $\mathcal{T}_i^{t-1}$*. By convention, the map* D_i *on the empty outcome path yields the empty set. Moreover, for analytical convenience, a death function is assumed to be such that* $D_i(\hat{h}^t) \cap \mathcal{B}_i^t = \emptyset$ *for all* $\hat{h}^t$*. In other words a death function cannot eliminate from the system any type which is being introduced into the system by the birth function at the same time. Finally, a death function cannot eliminate all existing types, so that* $\mathcal{D}_i^t$ *must be a strict subset of* $\mathcal{T}_i^t$ *for all* t*. The vector of death functions* $\{D_1, \ldots, D_N\}$ *is denoted by* D *for the rest of the paper.*

Combining the two definitions we have just given, we can define the 'birth rate' and the 'death rate' of the system at t in a straightforward way.

DEFINITION 7 *(Birth and Death Rates): Given a set of selection dynamics, a set of birth functions and a set of death functions, the birth rate of population* i *at time* t *is defined as* $b_i^t = \sum_{x \in \mathcal{B}_i^t} P_i^t(x)$*, whereas the death rate of population* i *at time* t *is defined as* $d_i^t = \sum_{x \in \mathcal{D}_i^t} P_i^t(x)$*. Finally, it is convenient to define the 'net normalized birth rate' of population* i *at time* t *as*

$$n_i^t = \frac{b_i^t - d_i^t}{1 - d_i^t} \tag{3}$$

We are now ready to put together the selection dynamics, the birth function, the death function and the information function which we have defined above. This yields the overall dynamics of our model. We combine these three elements with each other in an obvious way: $P_i^{t+1}(x)$ will be set equal to $(1 + g_i^t(x))P_i^t(x)$, and then appropriately 'normalized' to take into account the birth and death of types between t and $t+1$.

DEFINITION 8 *(Overall Dynamics): A vector of overall dynamics for the model is a quadruple* $F = \{G, B, D, Z\}$ *consisting of a vector of selection dynamics (*G*), a vector*

of birth functions (B), a vector of death functions (D), and an information function (Z). Given a vector of overall dynamics, an outcome path $\hat{h}^t$ and profile of probability distributions over types P^t, the profile of probability distributions P^{t+1} is defined by the following. For all $i = 1, \ldots, N$

$$P_i^{t+1}(x) = (1 - n_i^{t+1})(1 + g_i^t(x))P_i^t(x) \quad \text{if } x \in \mathcal{T}_i^t \text{ and } x \notin \mathcal{D}_i^{t+1} \tag{4}$$

where $g_i^t(x)$ is defined as in (1), and n_i^{t+1} is as in (3).[10] *Moreover, $P_i^{t+1}(x)$ is given by the birth function whenever $x \in \mathcal{B}_i^{t+1}$, and finally $P_i^{t+1}(x) = 0$ if $x \in \mathcal{D}_i^{t+1}$, and for all $x \notin \mathcal{T}_i^t \cup \mathcal{B}_i^{t+1}$.*

Given a vector of overall dynamics, F, and a Turing machine $x \in \mathbb{N}$, we denote the *birth date* of type x in population i by $t_i(x)$. Formally, $t_i(x)$ is the least t such that $x \in \mathcal{T}_i^t$. Notice that $t_i(x)$ is not defined if type x is never introduced in population i by the birth function B_i. Notice also that $P_i^{t_i(x)}(x)$ is the *birth probability* of x in population i.

At this point, it is useful to observe that, given a vector of overall dynamics, F, the entire history of the system (including frequencies) can be obtained by forward recursion through time in a way analogous to the outcome path $\hat{h}^t$ (see Section 3.1 above).

Given $\hat{h}^0$, the vector of birth functions B yields the vector of type sets $\mathcal{T}^0$, and the profile of probability distributions over types P^0. Given $\mathcal{T}^0$, the computations $\varphi_x(h^0)$ can be carried out for all $x \in \mathcal{T}_i^0$ and for all $i = 1, \ldots, N$. This yields the outcome path of length 1, $\hat{h}^1$, and the distribution of actions Q^0. Given $\hat{h}^1$, and Q^0, the selection dynamics and the vectors of birth and death functions now yield the profile of probability distributions over types P^1. The information function yields h^1 given $\hat{h}^1$, so that the actions of all types in $\mathcal{T}^1$ in period $t = 1$ can be computed next. Clearly, this procedure can be repeated any number of times to obtain the complete history of the system up to any arbitrary t.

[10] Given (3) and (2), it is immediate to check that $\sum_{x \in \mathcal{T}_i^t / \mathcal{D}_i^{t+1}} (1 - n_i^{t+1})(1 + g_i^t(x))P_i^t(x) = 1 - b_i^{t+1}$.

3.4. Computability Assumptions

Intuitively, our computability assumptions ensure that the forward recursion of the overall dynamics which we have just described in Section 3.3 is computable. Since some of the elements of the history of the system are, in general, vectors of *real* numbers, there are a number of ways to formulate a set of assumptions which are sufficient to ensure computability of the overall dynamics of the model. In this Section we describe the set of computability assumptions which we use for the rest of the paper. This is by no means the most general formulation possible. We use it purely for analytical convenience. Appendix D describes a much weaker set of computability assumptions under which the results of the paper are still valid.

In essence, the computability restrictions we work with, boil down to assuming that all probabilities in the history of the dynamical system are *rational numbers*, that all the components of the overall dynamics of the model are computable functions, and that the information function is computable.

We start with the formal Definition of a finite computable probability distribution and probability array over natural numbers.

DEFINITION 9 *(Computable Probability Distributions and Arrays): A finite computable probability distribution over the natural numbers is a finite subset of the naturals* $\{x_1, \ldots, x_K\}$ *and corresponding probabilities given by the rational numbers* $\{P(x_1), \ldots, P(x_K)\}$*, where* $P(x_i) > 0$ *for all* $i = 1, \ldots, K$ *and* $\sum_{i=1}^{K} P(x_i) = 1$*. A finite computable probability array is defined as a finite computable distribution without the constraint that probabilities must add up to one. Thus, for a finite computable probability array it may be the case that* $\sum_{i=1}^{K} P(x_i) < 1$*. Note that by Theorem A.1 a finite computable probability distribution (or array) can always be coded into a single natural number. With a slight abuse of notation, for the rest of the paper we will use the same symbol to indicate a finite computable probability distribution (or array) and its code in* $\mathbb{N}$*.*

Both finite computable probability arrays and the system's possible outcome paths (cf. Section 3.1 above) can be coded into the natural numbers. Therefore, it is now easy to state our assumption of computability of the vector of birth functions B.

ASSUMPTION 1 *(Computable Birth): For each population* $i = 1, \ldots, N$*, the birth function* B_i *is a computable from* $\mathbb{N}$ *to* $\mathbb{N}$*. Thus* $B_i(\hat{h}^t)$ *gives the code of the probability array representing the set of newborn types in population* i*,* $\mathcal{B}_i^t$ *and their rational probabilities* $\{P(x)\}_{x \in \mathcal{B}_i^t}$*.*

The assumption of computability of the vector of death functions takes the following form.

ASSUMPTION 2 *(Computable Death): For each population* $i = 1, \ldots, N$*, the death function* D_i *is a computable function from* $\mathbb{N}$ *to* $\mathbb{N}$*. Thus* $D_i(h^t)$ *gives the code of the subset* $\mathcal{D}_i^t$ *of existing types in* $\mathcal{T}_i^{t-1}$*, which disappear (have probability zero) at time* t*.*

Let a finite computable probability distribution over types P_i^t be given. By Definition 9, the corresponding probability distribution over actions, Q_i^t, is also a finite probability distribution over the naturals since each action in $\mathcal{A}_i$ can also be given a code in $\mathbb{N}$. This makes it possible to state our assumption of computable selection dynamics in the following form.

ASSUMPTION 3 *(Computable Selection Dynamics): For each population* $i = 1, \ldots, N$*, the selection dynamics* G_i *are given by a computable function from* $\mathbb{N}^{N+1}$ *to* $\mathbb{N}$*. Thus* $G_i[a_i^t(x), Q_1^t, \ldots, Q_N^t]$ *is the code of the (rational) growth rate of type* x *in population* i*, at time* t*.*

The last element of the model on which we place a computability assumption is the information function of Definition 2.

ASSUMPTION 4 *(Computable Information): The information function* Z *of Definition 2 is a computable function from* $\mathbb{N}$ *to* $\mathbb{N}$*.*

Assumption 4 is not sufficient for our global stability result to hold. The reason is that (as we explained intuitively in Section 1.3) we need our algorithmic types to be able to condition their actions on time at least. Neither Definition 2 nor Assumption 4, guarantee that the information h^t received at t, actually contains the date t. This is the reason for introducing our next assumption.

ASSUMPTION 5 *(Computable Time): The computable information function Z is such that the date can always be extracted computably from h^t. Formally, Z is such that there exists a total computable function T which satisfies $T(h^t) = t$ for all $h^t \in \mathcal{H}$.*

Throughout the rest of the paper, we call a vector of overall dynamics which satisfies all our computability assumptions a vector of computable overall dynamics.

We conclude this Section with an observation. Recall that in Section 3.3 we remarked that given the overall dynamics of the model, the complete history of the system up to any arbitrary finite date t can be obtained by forward induction through time. The computability requirements we have imposed on the system in this Section ensure that this property holds in a *computable* way for computable overall dynamics. It is useful to establish a name for a set of Turing machines which compute all the elements of a vector of computable overall dynamics.

DEFINITION 10 *(Basis): Consider a vector of overall dynamics, F, which satisfy the computability assumptions 1, 2, 3, 4 and 5. A vector of natural numbers $f \in \mathbb{N}^{3N+1}$ is called a 'basis' for the computable overall dynamics F if it is of the form $\{g_1, \ldots, g_N, b_1, \ldots, b_N, d_1, \ldots, d_N, z\}$, where g_i, b_i, d_i and z are Turing machines which compute G_i, B_i, D_i, and Z respectively. The set of $3N+1$-dimensional vectors of natural numbers which constitute a basis for some computable overall dynamics is denoted by $\mathcal{F}$ throughout the rest of the paper.*

We can now state formally the property of computable overall dynamics which we mentioned above.

REMARK 4 *(Forward Computable Dynamics): There exists a computable function S from $\mathbb{N}^{3N+2}$ to $\mathbb{N}$ with the following properties. Given any pair $\{f, t\}$ with $t \in \mathbb{N}$ and $f \in \mathcal{F}$ a basis for some computable overall dynamics F, $S(f, t)$ outputs (the code of) the pair $\{\hat{h}^t, P^t\}$ corresponding to F at t.*

PROOF: It is enough to notice that, by Theorem A.3 and by Church's Thesis, all the steps of the forward induction through time up to date t described at the end of

Section 3.3 can be performed by a Turing machine given the input pair $\{f, t\}$.

Q.E.D.

Note that since the proof of Remark 4 relies on a simulation argument, if f is not a basis for a vector of computable overall dynamics ($f \notin \mathcal{F}$) then the output of $S(f, t)$ need not be defined; for example, some (or all) of the elements of f could be machines which do not halt on some (or all) of the relevant inputs. Hence any attempt to simulate the dynamics forward would yield a computation which does not halt.

4. Stability Results

4.1. Preliminaries

Intuitively, there are two *minimal* conditions which we need to impose on the computable overall dynamics for the 'smart machine' argument which we outlined in Section 1.3 to work. Firstly, the rates of birth and death should not be too high so that the birth and death processes alone should not be able to determine the long-run behaviour of the dynamics, and hence prevent convergence. Secondly, the death process should be precluded from eliminating from the system the smart types: our stability results hinge on the presence of the smart machines.

To avoid the birth and death processes taking over the long-run behaviour of the system, it is enough to concentrate on the class of overall dynamics which guarantee that the sum over time of the birth and death rates in each population remains bounded. Throughout the rest of the paper we maintain the following assumption.

Assumption 6 *(Bounded Total Birth and Death): The overall dynamics, F, are assumed to have 'bounded total birth and death' in the sense that $\sum_{t=0}^{\infty} b_i^t < \infty$ and $\sum_{t=0}^{\infty} d_i^t < \infty$ for all $i = 1, \ldots, N$.*

To ensure that the death process does not interfere with the presence of smart types, for the rest of the paper we assume that, in any period, the top performers in terms of growth in each population are not eliminated by the death function. Since we are

not restricting attention to dynamics which are monotonic, or even payoff based, the property that the top performers should not be killed off by the death function is not stated with reference to the top payoff, but with reference to the top growth rate.

ASSUMPTION 7 *(Preservation of Top Performers): The overall dynamics, F, are assumed to 'preserve the top performers' in the sense that, for each population i, the death function D_i does not eliminate from the system at t any type which achieved the maximal growth rate in its population at time $t-1$. Formally, let*

$$\mathcal{G}_i^t = \arg\max_{x \in \mathcal{T}_i^t} g_i^t(x) \quad \forall\, i = 1, \ldots, N \;\; \forall\, t = 0, 1 \ldots$$

Then, the overall dynamics are assumed to be such that, for all $t = 0, 1, \ldots$, $\mathcal{D}_i^{t+1} \cap \mathcal{G}_i^t = \emptyset$, for all $i = 1, \ldots, N$.

It is convenient to parameterize the overall dynamics by the vector of type sets which are introduced into the system, through time, by the vector of birth functions.

DEFINITION 11 *(Support Sets): A vector of overall dynamics, F, is said to have the vector of support sets $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_N\}$ (or simply to have support $\mathcal{L}$) if and only if, for each population i, the birth function B_i introduces, through time, into the system a set of types equal to $\mathcal{L}_i \subseteq \mathcal{M}_i$. Formally, the overall dynamics F are said to have the vector of support sets $\mathcal{L}$, with $\mathcal{L}_i \subseteq \mathcal{M}_i$ for all $i = 1, \ldots, N$, if and only if $\mathcal{L}_i = \bigcup_{t=0}^{\infty} \mathcal{B}_i^t$, for all $i = 1, \ldots, N$. We will also say that the overall dynamics have support at least $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_N\}$ if and only if $\mathcal{L}_i \subseteq \bigcup_{t=0}^{\infty} \mathcal{B}_i^t$, for all $i = 1, \ldots, N$.*

The stability results which we present in Sections 4.2 and 4.3 below hinge on different versions of the smart machine argument which we outlined in Section 1.3. They are, in essence, different ways to ensure that the smart machines appear in the N populations at some point in time. The existence of a smart machine which takes an action that is optimal for population i for all t, is a preliminary result common to all our stability results. Therefore we present it here, before the actual theorems below.

Some further notation is needed: we denote by $\overline{g}_i^t$ the growth rate excluding birth and death in population i at time t which is maximal within the existing set of types,

$\mathcal{T}_i^t$. Thus,

$$\overline{g}_i^t = \max_{x \in \mathcal{T}_i^t} g_i^t(x) \quad \forall\, i =, 1, \ldots, N \;\; \forall\, t = 0, 1, \ldots \tag{5}$$

The growth rate $\overline{g}_i^t$ defined in (5) is maximal among the existing types. Since it could be that $\mathcal{T}_i^t$ does not contain a set of machines which take all possible different actions in $\mathcal{A}_i$ at t, it could be that $\overline{g}_i^t$ is not the maximal growth rate excluding birth and death which *any* machine could guarantee at t. We denote the latter by $\hat{g}_i^t$. Hence

$$\hat{g}_i^t = \max_{a_i \in \mathcal{A}_i} G_i\left[a_i, Q^t\right] \quad \forall\, i =, 1, \ldots, N \;\; \forall\, t = 0, 1, \ldots \tag{6}$$

Note that by definition $\hat{g}_i^t \geq \overline{g}_i^t$ for all i and t. Finally, we let

$$\overline{\mathcal{A}}_i^t \equiv \left\{a_i \in \mathcal{A}_i \;\mid\; G_i\left[a_i, Q^t\right] = \hat{g}_i^t\right\} \tag{7}$$

The following is a direct consequence of Remark 4, of the so-called 's-m-n' theorem (Theorem A.2) and of the existence of a 'universal Turing machine' (Theorem A.3)

Lemma 1 *(Existence of Smart Types): For all $i = 1, \ldots, N$ there exists a total computable function $K_i : \mathbb{N}^{3N+1} \to \mathbb{N}$ with the following properties. Let F be a vector of computable overall dynamics, and f be a basis for it. Then, $K_i(f)$ is the Gödel number of a Turing machine in $\mathcal{M}_i$ which, given any h^t, yields the action that maximizes growth at t excluding birth and death in population i. Formally, K_i is such that if $f \in \mathcal{F}$ is a basis for F, then*

$$G_i\left[\varphi_{K_i(f)}(h^t), Q^t\right] = \hat{g}_i^t \quad \forall\, t = 0, 1, \ldots \tag{8}$$

where h^t and Q^t are the information and distribution of actions at t generated by F.

Proof: See Appendix B.

Note that since the proof of Lemma 1 relies on the forward simulation argument which proves Remark 4, the output of $\varphi_{K_i(f)}(h^t)$ need not be defined if f is not a basis for a vector of computable overall dynamics.

We are now ready to present our global stability results. Theorems 1, 3 and 4 embody different sets of conditions which guarantee that all the population frequencies converge to a well defined limit. However, before we proceed, it is useful to emphasize the following point. Since each type in each population can take different actions at different times, clearly convergence of the population frequencies *does not* imply convergence of the action frequencies. We study the limit points of the action frequencies in Section 5, and find that under some additional conditions, they correspond to the Nash equilibria of the underlying game.

4.2. Global Stability with a 'Grain of Truth'

Consider a vector of computable overall dynamics F and let f be a basis for the system. Suppose further that for each population i, the birth function B_i is such that machine $K_i(f)$ is introduced into the system at some point in time (the birth date $t_i(K_i(f))$ is defined). Then the population frequencies must all converge to a well defined limit (Theorem 1 below). We call this result 'stability with a grain of truth', since the assumption that machine $K_i(f)$ must appear in F at some point in time has a flavour similar to the main result of Kalai and Lehrer (1993). The overall dynamics F must give positive probability of birth to a machine which behaves 'as if' it knew the true nature of system and used this computation to arrive at an optimal action at each t.

The intuition behind this result is simple to outline; the time path of the frequency of the smart machine is sufficiently close to being monotonically increasing[11] to 'pin down' the entire dynamics of the system. As we mentioned in Section 1.3 this can be viewed as an argument analogous to a Lyapunov stability result. A complementary way to understand intuitively what drives the result is the following. Lemmas B.1 and B.2 guarantee that if it is not the case that all population frequencies converge to a well defined limit, then the product $\prod_{t=0}^{\infty}(1+\overline{g}_i^t)(1-n_i^{t+1})$ diverges to infinity for some population i. Notice now that, provided $t_i(K_i(f))$ is defined, then it follows from $K_i(f)$ having maximal growth rate that the frequency in population i of machine

[11]Because of the birth and death processes, the frequency of a type which takes an action in $\overline{\mathcal{A}}_i^t$ for all t is not actually guaranteed to be monotonically increasing. See (4) above.

$K_i(f)$ at any $t > t_i(K_i(f))$ is given by $P_i^{t_i(K_i(f))}(K_i(f)) \cdot \prod_{\tau=t_i(K_i(f))}^{t}(1+\overline{g}_i^{\tau})(1-n_i^{\tau+1})$. It then follows that if some of the population frequencies do not converge to a well defined limit, for t large enough, the frequency of $K_i(f)$ would have to exceed one, giving an obvious contradiction.

THEOREM 1 *(Stability with a Grain of Truth): For any vector of computable overall dynamics F with support $\mathcal{L}$, there exists a vector of 'smart types' $\overline{x} = \{\overline{x}_1, \ldots, \overline{x}_N\}$ such that if $\overline{x} \in \mathcal{L}$, then all the population frequencies converge to a well defined limit in the sense that $\lim_{t\to\infty} P_i(x)$ exists for all $x \in \mathbb{N}$ and for all $i = 1, \ldots, N$. Moreover, the vector of smart types $\overline{x}$ can be effectively computed from a basis f for the overall dynamics F.*

PROOF: See Appendix B.

The proof of Theorem 1 involves setting $\overline{x}_i = K_i(f)$ for all $i = 1, \ldots, N$. Therefore, the following guarantees that Theorem 1 applies to a non-empty set of computable overall dynamics.

THEOREM 2 *(Existence with a Grain of Truth): There exists a vector of computable overall dynamics F such that, for some basis f for F, we have*

$$K_i(f) \in \mathcal{L}_i \quad \forall\, i = 1, \ldots, N \tag{9}$$

PROOF: The claim follows directly from the proof of Theorem 5 below.

Q.E.D.

4.3. Global Stability with a 'Provability' Restriction

The grain of truth assumption which drives Theorem 1 above is unsatisfactory since it is not a primitive assumption. As we define a vector of computable overall dynamics F, we implicitly define the corresponding vector $\overline{x}$ of smart types. If $\overline{x}$ happens to be in the support $\mathcal{L}$ of F, then Theorem 1 applies, otherwise it does not. Clearly, it would be desirable to find a set of *primitive* conditions on F which guarantee that the smart types are in the support of F and hence can drive convergence of the population frequencies as in Theorem 1.

Since the set of possible bases for computable overall dynamics is a countable one, it would seem tempting to proceed in the following way. For each f which is a basis for a vector of computable overall dynamics (for each $f \in \mathcal{F}$), define the vector of smart types $\{\overline{x}_1, \ldots, \overline{x}_N\} = \{K_1(f), \ldots, K_N(f)\}$. If we consider all possible bases we then get a countable set of possible smart types. One could then assume that the support of the computable overall dynamics should contain the *entire set* of possible smart types which we have just defined. Clearly, Theorem 1 would apply to any such computable overall dynamics. Unfortunately, this way of proceeding is flawed for the following reason. The set of all possible bases for computable overall dynamics lacks a 'regularity condition' known as 'recursive enumerability' (see Definition A.1); its elements cannot be exhaustively listed in a computable way. As a consequence, the set of possible smart types is not recursively enumerable either. On the other hand, the computability of the birth functions directly implies that the support sets of any computable overall dynamics *are* recursively enumerable (henceforth abbreviated r.e.) sets.

It is useful to state the above three facts formally for future reference.

REMARK 5 *(Set of Bases): The set $\mathcal{F}$ of $3N+1$-tuples of natural numbers which constitute a basis for computable overall dynamics as in Definition 10 is not r.e. according to Definition A.1.*

PROOF: See Appendix B.

REMARK 6 *(Set of Smart Types): The set of all possible smart types is not r.e. in the sense that for all $i = 1, \ldots, N$ the image under K_i of the set $\mathcal{F}$ of all possible bases of computable overall dynamics is not r.e. according to Definition A.1.*

PROOF: See Appendix B.

REMARK 7 *(Recursively Enumerable Support Sets): Given any vector of computable overall dynamics F, for all $i = 1, \ldots, N$ the support set $\mathcal{L}_i$ is r.e. according to Definition A.1. It follows from Remark 6 that there does not exist a computable overall dynamics with support sets equal to the sets of all possible smart types.*

PROOF: See Appendix B.

Two possible ways to overcome the 'regularity problem' we have outlined naturally come to mind. The first is to set the support of the overall dynamics equal to the set of all possible Turing machines. This is easily seen to be impossible since we have ruled out from our allowable algorithms (Definition 3) any non-halting Turing machines.[12]

The second is to set the support of the overall dynamics equal to the sets of allowable algorithms of Definition 3. The problem with the latter possibility is that the set of allowable algorithms is not r.e. since the set of Turing machines which compute a total function is not r.e.[13]

REMARK 8 *(Non-Halting and Allowable Algorithms): There does not exist a computable overall dynamics with support sets equal to the set of all possible Turing machines. Moreover, for any* $i = 1, \ldots, N$*, the set of allowable algorithms is not r.e. It follows that there does not exist a computable overall dynamics with support sets equal to the sets of allowable algorithms for each population (*$\mathcal{L}_i = \mathcal{M}_i$ *for all* i*).*

PROOF: See Appendix B.

The last possibility to overcome the regularity problem above, without resorting to further assumptions, would be to find (at least) one vector of computable overall dynamics with support sets which, for all $i = 1, \ldots, \mathbb{N}$, are strictly contained in the set of allowable algorithms, and which strictly contain the set of all possible smart

[12] Changing the definition of allowable algorithms to allow *all* possible Turing machines as admissible types would make it impossible to simulate the system forward as in Remark 4 since the simulation of a non-halting Turing machine does not halt in general (this is the so-called 'halting' problem for Turing machines (Cutland 1980, Ch.6)). In a model of repeated games, Anderlini and Sabourian (1995) allow *some* non-halting Turing machines as admissible types, but proceed to exclude them from any simulation of the model which is required. Introducing non-halting machines exactly as in Anderlini and Sabourian (1995) is nt possible here since the 'simulation requirements' are different between the two models ('finite' forward simulation is sufficient in Anderlini and Sabourian (1995) but it is not sufficient here). It is clear, however, that the nature of the regularity problem we have identified would clearly change if we did not insist on ruling out all non-halting algorithms from the set of allowable types. Therefore, it is possible that different formulations of our basic model, which treat differently non-halting Turing machines, may behave substantially differently from the present one in relation to the regularity problem above.

[13] The fact that the set on Turing machines which compute a total function is not r.e. is in turn a direct consequence of the Rice-Shapiro theorem (Theorem A.9).

types of Remark 6. We have attempted to show the existence of such dynamics and, as a result, we conjecture that such construction is impossible.[14]

Overcoming the regularity problem above, is sufficient to obtain stability by introducing all possible smart types in the support of the computable overall dynamics. This is confirmed by our next theorem; if one is willing to restrict attention to an r.e. *subset* of the set of bases $\mathcal{F}$, then it is possible to find a corresponding vector of *recursively enumerable* sets of smart types which guarantees stability of the computable overall dynamics.

Consider any r.e. subset $\mathcal{R}$ of the set of bases $\mathcal{F}$. Since $\mathcal{R}$ is r.e. we can find a $3N+1$-tuple $\{q_1, \ldots, q_{3N+1}\}$ of Turing machines which enumerates it as in Theorem A.4. For all $i = 1, \ldots, N$ now set

$$\mathcal{L}_i(\mathcal{R}) = \operatorname*{Range}_{v \in \mathbb{N}} K_i(\varphi_{q_1}(v), \ldots, \varphi_{q_{3N+1}}(v)) \tag{10}$$

where K_i is the total computable function of Lemma 1. Thus, $\mathcal{L}_i(\mathcal{R})$ is the set of smart machines for population i corresponding to the set of bases $\mathcal{R} \subset \mathcal{F}$. Note that $\mathcal{L}_i(\mathcal{R})$ is an r.e. set by Theorem A.6 and let $\mathcal{L}(\mathcal{R}) = \{\mathcal{L}_i(\mathcal{R}), \ldots, \mathcal{L}_N(\mathcal{R})\}$.

Consider any subset $\mathcal{R}$ of $\mathcal{F}$. Let $\Phi(\mathcal{R})$ represent the set of overall dynamics corresponding to the set of bases $\mathcal{R}$. In other words, given any $\mathcal{R} \subseteq \mathcal{F}$,

$$F \in \Phi(\mathcal{R}) \quad \Leftrightarrow \quad \exists\, f \in \mathcal{R} \text{ with } f \text{ a basis for } F$$

We are now ready to state our next result.

THEOREM 3 *(Stability with R.E. Bases): Consider any recursively enumerable set of bases $\mathcal{R} \subset \mathcal{F}$. Suppose that $F \in \Phi(\mathcal{R})$ has support at least $\mathcal{L}(\mathcal{R})$. Then all the population frequencies corresponding to such F converge to a well defined limit in the sense that $\lim_{t\to\infty} P_i^t(x)$ exists for all i and for all $x \in \mathbb{N}$.*

[14] While we do not have a proof of this claim, it is clear that it would be consistent with what is known from the computability literature. In the jargon of recursive function theory, we conjecture that, for all $i = 1, \ldots, N$, the complement in $\mathbb{N}$ of the set of allowable algorithms, and the set of all possible smart types are 'effectively inseparable'. (See, for instance, Cutland (1980) Ch.7 and particularly exercise 7.3.13(9), or Rogers (1967) exercise 2.30 and Ch.7, particularly Definition 7.7.4.)

PROOF: See Appendix B.

Before proceeding any further, it is helpful to emphasize the following point.

REMARK 9 *(Lower Bounds): Recall that in Definition 11 of support sets we say that F has support at least $\mathcal{L}$, if $\mathcal{L}$ is a vector of 'lower bounds' on the type sets which are introduced into the system by the vector of birth functions. It follows that Theorem 3 guarantees stability of any computable overall dynamics $F \in \Phi(\mathcal{R})$ with support $\mathcal{L}'$ such that $\mathcal{L}_i(\mathcal{R}) \subseteq \mathcal{L}'_i$ for all $i = 1, \ldots, N$, which satisfy the other hypotheses of the Theorem. Finally, note that the observation we have just made about Theorem 3 applies, mutatis mutandis, to Theorem 4 below.*

Theorem 3 applies to any r.e. set of bases $\mathcal{R} \subset \mathcal{F}$. One such set is the set of $f \in \mathcal{F}$ for which a 'proof' of the fact that f is indeed a basis for a set of computable overall dynamics is available. The notion of 'provability', of course, has to be treated with care for this line of argument to be meaningful. In Appendix C we provide an outline of the relevant definitions and results. The key feature of provability which Theorem 4 uses is that if a proof exists that $f \in \mathcal{F}$, then it must be possible to check, computably, that this is indeed the case. This gives us the required recursive enumerability of the set of bases we use in the theorem.

THEOREM 4 *(Stability with Provability): Let a Recursively Axiomatized Formal System, Λ, be given (see Definition C.2). Assume that the class of statements $f \in \mathcal{F}$ is representable in Λ (see Definition C.4). Let $\mathcal{R}$ be the set of $f \in \mathbb{N}^{3N+1}$ such that the statement $f \in \mathcal{F}$ is provable within Λ (see Definition C.2). Then all the population frequencies of any computable overall dynamics $F \in \Phi(\mathcal{R})$ which has support at least $\mathcal{L}(\mathcal{R})$, converge to a well defined limit in the sense that $\lim_{t\to\infty} P_i^t(x)$ exists for all i and for all $x \in \mathbb{N}$.*

PROOF: By Theorem C.3, $\mathcal{R}$ as defined is an r.e. set contained in $\mathcal{F}$. Hence Theorem 3 applies directly to establish the result.

Q.E.D.

The following shows that Theorem 4 applies to a non-empty set of computable overall dynamics when Λ is chosen to be the formal system which is most 'natural' in the present context.

THEOREM 5 *(Existence with Provability): Let Λ^* be the Recursively Axiomatized Formal System known as Formal Number Theory (see Definition C.3 and Theorem C.2). Let $\mathcal{R}^*$ be the set of bases such that the statement $f \in \mathcal{F}$ is provable within Λ^*, and note that $\mathcal{R}^*$ is recursively enumerable by Lemma C.2. Then there exists a set of Computable Overall Dynamics $F^* \in \Phi(\mathcal{R}^*)$ which has support at least $\mathcal{L}(\mathcal{R}^*)$.*

PROOF: See Appendix C.

The proof of Theorem 5 is rather lengthy, but not difficult to outline intuitively. The proof is constructive and displays a vector of overall dynamics which satisfies the requirements of the Theorem, and which, for simplicity, is taken to have a trivial death process which does not kill off any existing types at any time. For simplicity again, the selection dynamics of the overall dynamics constructed in the proof are taken to be the 'replicator dynamics' of Definition 14 below. Given that the set $\mathcal{R}^*$ of the statement of Theorem 5 is r.e., it is not hard to show that the sets of smart types corresponding to $\mathcal{R}^*$, namely $\mathcal{L}_i(\mathcal{R}^*)$, are also r.e. The next step in the proof is to show that, for each i, there exists a Turing machine b_i^* which enumerates $\mathcal{L}_i(\mathcal{R}^*)$ (as in Theorem A.5), and which is such that there exists a proof within Formal Number Theory of the fact that b_i^* enumerates the set $\mathcal{L}_i(\mathcal{R}^*)$. This step is feasible since the procedure which yields each b_i^* is entirely constructive. The proof is then concluded setting the birth process of the overall dynamics to be just the enumeration of $\mathcal{L}_i(\mathcal{R}^*)$ given by b_i^* for each $i = 1, \ldots, N$. The only type born in each population i at time t is precisely the t-th element in the enumeration of $\mathcal{L}_i(\mathcal{R}^*)$ given by b_i^*, and its birth probability is set to $1/2^t$.

Once Theorem 5 is established, using Remark 9 and Theorem A.7 it is immediate to see that Theorem 4 applies to a countable infinity of different computable overall dynamics. We state the following without proof.

COROLLARY 1: *Let Λ^*, $\mathcal{R}^*$ and $\Phi(\mathcal{R}^*)$ be as in Theorem 5. Then there exists a countable infinity of distinct vectors of computable overall dynamics F such that $F \in \Phi(\mathcal{R}^*)$ and F has support at least $\mathcal{L}(\mathcal{R}^*)$.*

5. The Limit Points of the Action Frequencies

5.1. Continuous, Payoff-Based, Monotonic and Replicator Dynamics

The definition of selection dynamics which we have given in Section 3.3 is very general; it encompasses an extremely wide variety of possible selection dynamics. For instance, it is consistent with a system which rewards those types which do *badly* in terms of payoffs. In fact, (1) is clearly consistent with the widest possible range of selection dynamics which depend on the distribution of *payoffs* in each period. To see this, notice that we can define the expected (or average) payoff to type x in population i at time t as

$$\Pi_i^t(x) \equiv E_{Q_{-i}^t}\left\{ \pi_i \left[a_i^t(x)\,,\, a_{-i}\right] \right\}$$

where the expectation is taken over $a_{-i} \equiv \{a_1, \ldots, a_{i-1}\ a_{i+1}, \ldots, a_N\}$ according [15] to the probability distribution $Q_{-i}^t \equiv \{Q_1^t, \ldots, Q_{i-1}^t\ Q_{i+1}^t, \ldots, Q_n^t\}$. Given a profile of probability distributions over actions at t, it is possible to define the probability distribution over payoffs in population i at time t; we denote the latter by Π_i^t, and the profile of probability distributions over payoffs at t by $\Pi^t \equiv \{\Pi_1^t, \ldots, \Pi_N^t\}$.

In order of decreasing generality, two classes of selection dynamics which are consistent with (1), are as follows.

Definition 12 *(Payoff-Based Dynamics): The selection dynamics G are said to be payoff-based dynamics if and only if the vector of functions G can be written in the following form: for all $i = 1, \ldots, N$*

$$G_i\left[a_i^t(x), Q^t\right] = \hat{G}_i\left[\Pi_i^t(x), \Pi^t\right] \quad \forall\, x \in \mathcal{T}_i^t \quad \forall\, t = 0, 1, \ldots N$$

Definition 13 *(Monotonic Dynamics): The selection dynamics G are said to be monotonic in payoffs (or simply monotonic) if and only if they are payoff-based and they satisfy*

$$\Pi_i^t(x) > \Pi_i^t(x') \Rightarrow \hat{G}_i\left[\Pi_i^t(x), \Pi^t\right] > \hat{G}_i\left[\Pi_i^t(x'), \Pi^t\right] \quad \forall\, x \in \mathcal{T}_i^t \ \ \forall\, t = 0, 1, \ldots$$

[15] Here, and throughout the rest of the paper, we follow the standard notation of denoting by a subscript of $-i$ a vector without its i-th component.

for all $i = 1, \ldots, N$

A special case of monotonic dynamics is the 'replicator dynamics' which have been extensively studied in previous literature (VanDamme 1987, Hofbauer and Sigmund 1988, Weibull 1992, among others). One possible definition of replicator dynamics, which we state for future reference, is the following.

DEFINITION 14 *(Replicator Dynamics): The selection Dynamics are said to be the replicator dynamics if and only if the vector of growth functions* G *takes the following form*

$$G_i\left[a_i^t(x), Q^t\right] = \lambda\left[\Pi_i^t(x) - \sum_{x' \in \mathcal{T}_i^t} \Pi_i^t(x') P_i^t(x')\right] \quad \forall\, x \in \mathcal{T}_i^t \quad \forall\, i = 1, \ldots, N \qquad (11)$$

where λ *is a 'small' positive constant which ensures that the dynamics do not yield 'negative probabilities' at any time*

One further class of selection dynamics in which we will be interested below is the following.

DEFINITION 15 *(Continuous Dynamics): The selection dynamics* G *are said to be continuous if and only if, for all* i *and for any given* $a_i \in \mathcal{A}_i$, G_i *is continuous in* Q, *for all* $Q \in \Delta^A$.

We conclude this section by noting that the replicator dynamics we have just defined are clearly continuous.

5.2. Nash Equilibria

The second main result of this paper is that, under certain additional conditions, the presence of smart types in the support of the overall dynamics not only guarantees convergence of the population frequencies, but it also implies that *all the limit points* of the frequencies of actions in each population will coincide with the Nash equilibria (pure or mixed) of the underlying game Γ.

The next remark clarifies why we are referring to all the limit points of Q^t rather than simply to *the* limit of Q^t.

REMARK 10: *Because types in our model can take different actions at different dates, convergence of the population frequencies is not sufficient to guarantee convergence of the sequence* Q^t.

Besides those which guarantee convergence of the population frequencies, to guarantee convergence to Nash[16] the selection dynamics of the model must be continuous and monotonic (see Definitions 15 and 13).

We are now ready to state formally our second main result.

THEOREM 6 *(Nash Equilibria): Suppose that the Computable Overall Dynamics,* F*, are such that the selection dynamics are continuous and monotonic. Let* f *be a basis for* F*. Suppose also that* F *is such that for all* $i = 1, \ldots, N$*, the smart type* $K_i(f)$ *of Lemma 1 is in the support* $\mathcal{L}_i$ *of population* i*. Then, all the limit points of* Q^t *are Nash equilibria (pure or mixed) of the underlying game* Γ.

PROOF: See Appendix B.

It is worth dwelling on the statement of Theorem 6 by means of a diagram. For illustrative purposes only, imagine that Γ is a two-by-two game so that the action frequencies in each population can be measured by a single number. Imagine also that Γ has multiple Nash equilibria.

In Figure 1 we measure the action frequencies in population i on the vertical axis, with NE_1 representing one Nash equilibrium, and NE_2 representing another Nash equilibrium of Γ. On the horizontal axis we measure time. The points in the diagram, depict a possible sequence of Q_i^t which does not converge, but which has

[16] As we mentioned in Section 1.3, these conditions are analogous to those found in several previous contributions (Nachbar 1990, Samuelson and Zhang 1992, among others).

two subsequences, one converging to NE_1, and the other converging to NE_2.

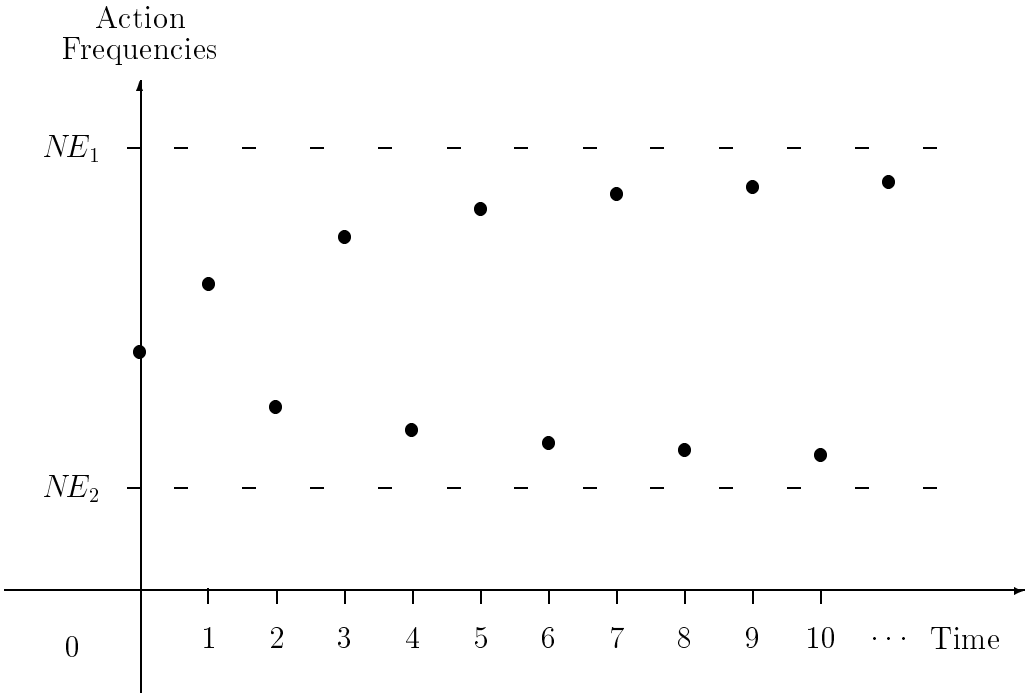

Figure 1

We conclude this section with an observation.

Remark 11 *(Unique Nash Equilibrium): Under the hypotheses of Theorem 6 every convergent subsequence of Q^t will converge to a Nash equilibrium (pure or mixed) of the underlying game Γ. It naturally follows that if Γ has a unique Nash equilibrium (pure or mixed), then the sequence Q^t converges to such equilibrium.*

6. Concluding Remarks

6.1. Alternative Formulations and Extensions

It is worth mentioning one possible extension of our model, and an alternative way to formulate the model itself.

In Remark 2 above, we emphasized that information is implicitly assumed to be symmetric in our model. This is because all types receive the same information at

t, namely $Z(\hat{h}^t)$. It is not difficult to see how this assumption can be relaxed to allow for asymmetric information, without affecting the main results of the paper. Suppose that the information which type x in population i receives at t were written as $Z_i(x, \hat{h}^t)$, so that dependence on type and population is allowed. Suppose also that Z_i is a computable function of both it arguments. Then clearly, it would still be possible to simulate any vector of computable overall dynamics by forward recursion as in Remark 4; it follows that all the main results of the paper can be extended to this case with relatively small changes.

We have formulated our model insisting that only a *finite* set of types be actually present in the system at any time t; although the birth function can make the support sets of the overall dynamics actually infinite. This makes the forward simulation of any computable overall dynamics a relatively straightforward matter. At the expense of some considerable complication, using some of the techniques described in Appendix D, we could have formulated our model as an infinite one; overall dynamics with an infinite set of types in the initial distribution and no birth or death through time. This would make our model closer to the standard[17] evolutionary model. It would however, complicate considerably the forward simulation of any computable overall dynamics. This is simply because, with an infinite set of types present from the start, it would be impossible to simulate the *entire* model forward. Any finite approximation to the true model can still be simulated, however. Including more and more of the infinite set of types in the finite approximation would then make the simulation more and more precise, and in the limit, exact. It follows that by treating the required levels of approximation with sufficient care, the (approximately) 'smart types' can be constructed for the infinite model as well. It then follows relatively easily, that all the main results of this paper can be reformulated and hold for a model of computable dynamics with an infinite initial distribution which includes an appropriately 'rich' set of types.[18]

[17] See footnote 2 above.

[18] Earlier versions of this paper analysed a model with an infinite initial distribution and no birth or death.

6.2. The Role of Computability

We conclude the paper with a further discussion of the role which the assumptions of computability play in our stability[19] results.

In Section 1.3 above, we mentioned the three features of the computability framework which are essential to the proofs of our results: a countable set of possible types, the existence of the universal programme, and the s-m-n parameterization theorem. At a more general level, in our view there are two crucial consequences of the computability assumptions which we have made above. The first is that restricting attention to algorithmic types and computable dynamics (selection, birth, death and information), makes the model automatically a 'closed' model in the following sense.

Suppose that instead of considering the set of allowable algorithms of Definition 3, we considered an arbitrary set of functions (maps from information into actions) as our possible types. Suppose also that instead of restricting attention to computable dynamics we allowed for an arbitrary class of dynamical systems. We could then attempt to replicate the simplest of our convergence results (the grain of truth result of Theorem 1) by constructing the vector of smart types $K_i(f)$ of Lemma 1 for the given dynamical system identified by f, and then assuming that these types are in the support of the system itself. The difficulty would be that the range of the function K_i may well not be contained in the arbitrary set of types which we started off with making the operation impossible. Our computability assumptions guarantee that the model is closed in the sense that staring with computable types, the smart types which we construct are always guaranteed to be precisely computable maps from information into actions, and hence allowable types.

An obviously tempting route to avoid having to make any computability assumptions and force the model to be closed in the sense we have described would be to allow for the largest possible set of types in the first place. However, the set of all maps from information into actions has the cardinality of the continuum. This may make the set of possible smart types a continuum as well. In turn, this would preclude the way to any stability result of the 'large support' variety (as opposed to the grain

[19]It is clear that Theorem 6 above does *not* depend on our computability assumptions.

of truth variety) since it would be impossible to give an atom of probability to all possible smart types.

The second crucial consequence of the computability framework is to make the provability restriction of Theorems 4 and 5 a 'natural' one to consider. This plays an essential role in the possibility of establishing our 'large support' stability results. As we mentioned in Section 4.3, as soon as we attempt to construct a vector of computable overall dynamics which has a support that contains all possible smart types, we run up against the following difficulty: the support of any computable overall dynamics is r.e., while the set of all possible smart types is not r.e. Restricting attention to those dynamical systems which satisfy our provability restriction solves the problem since it makes the set of corresponding possible smart types an r.e. set.

The provability restriction *jointly* with our computability assumptions make it possible to find a class of dynamical systems which simultaneously satisfy the two requirements of being closed in the sense we have outlined, and of containing in their support all the relevant possible smart types.

APPENDIX A

We start with some results which are standard in the literature on recursive function theory. All the results which we state without proof can be found, for instance, in Davis (1958), Rogers (1967) or Cutland (1980).

THEOREM A.1 *(Gödel Numbering): Let a fixed set of symbols* $\mathcal{S}^0 = \{S_0^0, S_1^0, \ldots, S_i^0, \ldots\}$ *be given. Consider the set of finite 'first order' strings of symbols of the type* $S^1 = \{S_{i_1}^0, \ldots, S_{i_K}^0\}$ *with* $K \geq 2$*. Let this be denoted by* $\mathcal{S}^1$*. Next, by induction, define the set of finite 'n-th order strings' for any* $n \in \mathbb{N}$ *as follows. Given the set of finite 'n−1-th order' strings,* $\mathcal{S}^{n-1}$ *with typical element* S_i^{n-1}*, define the set of finite 'n-th order' strings,* $\mathcal{S}^n$*, as the set of objects of the type* $\{S_{i_1}^{n-1}, \ldots, S_{i_K}^{n-1}\}$ *with* $K \geq 2$*, with typical element* S_i^n*. Finally, let the set of finite strings of any order,* $\mathcal{S}$*, be defined as* $\bigcup_{n\in\mathbb{N}} \mathcal{S}^n$*. Then there exists a one-to-one total computable function (the 'coding' function)* $C : \mathcal{S} \to \mathbb{N}$ *which assigns a unique code in* $\mathbb{N}$ *to each element of* $\mathcal{S}$*. Moreover, C can be inverted in a computable way in the sense that there exists a one-to-one total computable function (the 'decoding' function)* $D : \mathbb{N} \to \mathcal{S}$ *such that* $D(C(S)) = S$ *for all* $S \in \mathcal{S}$*.*

THEOREM A.2 *(s-m-n): For each* $m \geq 0$ *and* $n \geq 1$ *there exists a total computable function of* $m+1$ *variables,* W*, such that* $\forall\, e \in \mathbb{N}$ *and* $\forall\, \{h_1, \cdots, h_m, h_{m+1}, \cdots, h_{m+n}\} \in \mathbb{N}^{m+n}$ *we have*

$$\varphi_e(h_1, \cdots, h_{m+n}) \simeq \varphi_{W(e,h_1,\cdots,h_m)}(h_{m+1}, \cdots, h_{m+n})$$

THEOREM A.3 *(Universal Turing Machine): Given any* $m \geq 1$*, there exists a number* u_m*, such that*

$$\varphi_{u_m}(n, e_1, \cdots, e_m) \simeq \varphi_n(e_1, \cdots, e_m) \;\; \forall\, \{n, e_1, \cdots, e_m\} \in \mathbb{N}^{m+1}$$

DEFINITION A.1 *(Recursively Enumerable Set): A set* $\mathcal{S} \subseteq \mathbb{N}^m$ *is recursively enumerable (abbreviated r.e.) if and only if it is equal to the domain of a computable function of m variables. Formally,* $\mathcal{S} \subseteq \mathbb{N}^m$ *is r.e. if and only if for some* $k \in \mathbb{N}$ *we have* $\varphi_k(e_1, \ldots, e_m) \downarrow \Leftrightarrow \{e_1, \ldots, e_m\} \in S$*. (The empty set is r.e. since the function 'nowhere defined' is computable.)*

THEOREM A.4: *A non-empty set $\mathcal{S} \subseteq \mathbb{N}^m$ is r.e. if and only if it is the range of an m-tuple of total computable functions. Formally $\mathcal{S} \subseteq \mathbb{N}^m$ is r.e. if and only if there exists an m-tuple of Turing machines $\{k_1, \ldots, k_m\}$ computing total computable functions such that*

$$\{e_1, \ldots, e_m\} \in \mathcal{S} \quad \Leftrightarrow \quad \exists v \text{ such that } \varphi_{k_1}(v) = e_1, \ldots, \varphi_{k_m}(v) = e_m \tag{A.1}$$

Given an r.e. set $\mathcal{S}$, an m-tuple of Turing machines $\{k_1, \ldots, k_m\}$ with the property in (A.1) is said to 'enumerate' $\mathcal{S}$. We refer to $\{\varphi_{k_1}(v), \ldots, \varphi_{k_m}(v)\}$ as the v-th element in the enumeration of $\mathcal{S}$.

THEOREM A.5: *An infinite set $\mathcal{S} \subseteq \mathbb{N}^m$ is r.e. if and only if it is the range of an m-tuple of Turing machines $\{k_1, \ldots, k_m\}$ as in Theorem A.4, with the additional property that*

$$v \neq v' \Rightarrow \{\varphi_{k_1}(v), \ldots, \varphi_{k_m}(v)\} \neq \{\varphi_{k_1}(v'), \ldots, \varphi_{k_m}(v')\}$$

Turing machines $\{k_1, \ldots, k_m\}$ are said to enumerate $\mathcal{S}$ 'without repetitions'.

THEOREM A.6: *The range of a computable function of m variables is r.e.*

THEOREM A.7: *The intersection of two r.e. sets is r.e. The union of two r.e. sets is r.e.*

DEFINITION A.2 *(Recursive Set): A set $\mathcal{S} \subseteq \mathbb{N}^m$ is recursive if and only if it is recursively enumerable and its complement is recursively enumerable.*

DEFINITION A.3 *(Finite Function): A partial function C from $\mathbb{N}$ to $\mathbb{N}$ is said to be a finite function if and only if $C(e)$ is defined only for a finite set of values of e.*

THEOREM A.8: *Any finite function is computable.*

THEOREM A.9 *(Rice-Shapiro): Let $\boldsymbol{\mathcal{C}}$ be a set of computable functions from $\mathbb{N}$ to $\mathbb{N}$ such that the set*

$$\mathcal{C} = \{c \in \mathbb{N} \mid \exists C \in \boldsymbol{\mathcal{C}} \text{ such that } \varphi_c(e) \simeq C(e) \ \forall\, e \in \mathbb{N}\} \tag{A.2}$$

is recursively enumerable. Then $C \in \boldsymbol{\mathcal{C}}$ if and only if there exists a finite function $C' \in \boldsymbol{\mathcal{C}}$ such that $C'(e)\downarrow \Rightarrow C'(e) = C(e)$.

APPENDIX B

PROOF OF LEMMA 1: Using Theorem A.3, we start by constructing a function C_i from $\mathbb{N}^{3N+2}$ to $\mathbb{N}$ as follows. Given an input pair $\{f, h^t\}$, the function C_i is described by the following sequence of steps. First of all compute $T(h^t)$, where T is the computable function of Assumption 5 (Computable Time), so that $T(h^t) = t$. After this computation is completed, C_i computes the output of $S(f,t)$, where S is the computable function of Remark 4, so that $S(f,t)$ is (the code of) the pair $\{\hat{h}^t, P^t\}$. Given P^t it is now feasible to compute Q^t. Using Assumption 3 (Computable Selection Dynamics), C_i can now go on to compute the growth rates $G_i[a_i, Q^t]$ for all $a_i \in \mathcal{A}_i$. Picking out the actions which give the maximal growth rates now yields the set $\overline{\mathcal{A}}_i^t$. Finally, C_i outputs an arbitrarily chosen element of $\overline{\mathcal{A}}_i^t$. Therefore we have that if $f \in \mathcal{F}$, then for all $i = 1, \ldots, N$, and for all h^t

$$C_i(f, h^t) \in \overline{\mathcal{A}}_i^t \tag{B.1}$$

Since C_i is a computable function, for each i there exists a Turing machine c_i such that $\varphi_{c_i}(f, h^t) \simeq C_i(f, h^t)$. We can now apply Theorem A.2 (*s-m-n*) to obtain a total computable function $W : \mathbb{N}^{3N+2} \to \mathbb{N}$ such that $\varphi_{W(c_i,f)}(h^t) \simeq \varphi_{c_i}(f, h^t) \simeq C_i(f, h^t)$. Setting $K_i(f) \simeq W(c_i, f)$ finally yields

$$\varphi_{K_i(f)}(h^t) \simeq C_i(f, h^t) \quad \forall\, f \;\; \forall\, h^t \tag{B.2}$$

This, together with (B.1), is clearly enough to prove the claim.

Q.E.D.

LEMMA B.1: *Let a vector of overall dynamics, F, be given. For all $i = 1, \ldots, N$, if there exists an $x \in \mathbb{N}$ such that $\lim_{t\to\infty} P_i^t(x)$ does not exist, then $\sum_{t=0}^{\infty} \overline{g}_i^t = +\infty$.*

PROOF: By (5), we have

$$P_i^t(x) g_i^t(x) \leq P_i^t(x) \overline{g}_i^t \leq \overline{g}_i^t \quad \forall\, x \in \mathcal{T}_i^t \;\; \forall\, t = 0, 1, \ldots \;\; \forall\, i = 1, \ldots, N \tag{B.3}$$

Suppose now that, for some i, t, and $x \in \mathcal{T}_i^t$

$$P_i^t(x) g_i^t(x) < -\overline{g}_i^t \tag{B.4}$$

then, since $\sum_{x\in\mathcal{T}_i^t} P_i^t(x) = 1$ and $\overline{g}_i^t \geq 0$, we would have that

$$P_i^t(x)g_i^t(x) < -\overline{g}_i^t \sum_{x'\in\mathcal{T}_i^t/x} P_i^t(x')$$

Therefore, using again the fact that $\overline{g}_i^t$ is maximal,

$$P_i^t(x)g_i^t(x) < -\sum_{x'\in\mathcal{T}_i^t/x} P_i^t(x')g_i^t(x')$$

and hence

$$\sum_{x\in\mathcal{T}_i^t} P_i^t(x)g_i^t(x) < 0 \tag{B.5}$$

Since (2) must hold, (B.5) is a contradiction. We can then conclude that (B.4) must be false. Using this fact and (B.3), we have that

$$\left|P_i^t(x)g_i^t(x)\right| \leq \overline{g}_i^t \quad \forall\, x \in \mathcal{T}_i^t \;\; \forall\, t = 0, 1, \ldots \;\; \forall\, i = 1, \ldots, N \tag{B.6}$$

Since $-1 \leq P_i^t(x)g_i^t(x) \leq 1$, using (4) we get

$$\left|P_i^{t+1}(x) - (1 + g_i^t(x))P_i^t(x)\right| \leq \left|n_i^{t+1}\right| \quad \forall\, x \in \mathcal{T}_i^t \;\; \forall\, t = 0, 1, \ldots \;\; \forall\, i = 1, \ldots, N \tag{B.7}$$

By the triangular inequality, we also have

$$\left|P_i^{t+1}(x) - P_i^t(x)\right| \leq \left|g_i^t(x)P_i^t(x)\right| + \left|P_i^{t+1}(x) - (1 + g_i^t(x))P_i^t(x)\right| \tag{B.8}$$

Finally, substituting (B.6) and (B.7) into (B.8) we obtain

$$\left|P_i^{t+1}(x) - P_i^t(x)\right| \leq \overline{g}_i^t + \left|n_i^{t+1}\right| \quad \forall\, x \in \mathcal{T}_i^t \;\; \forall\, t = 0, 1, \ldots \;\; \forall\, i = 1, \ldots, N \tag{B.9}$$

Simple algebra shows that $|n_i^t| \leq b_i^{t+1} + d_i^{t+1}/(1 - d_i^{t+1})$, for all i and t. Moreover, for any $x \notin \mathcal{T}_i^t$, by definition of b_i^{t+1}, we must have $\left|P_i^{t+1}(x) - P_i^t(x)\right| \leq b_i^{t+1}$. Therefore, using inequality (B.9) and the fact that by definition $\overline{g}_i^t \geq 0$, we have

$$\left|P_i^{t+1}(x) - P_i^t(x)\right| \leq \overline{g}_i^t + b_i^{t+1} + \frac{d_i^{t+1}}{1 - d_i^{t+1}} \quad \forall\, x \in \mathbb{N} \;\; \forall\, t = 0, 1, \ldots \;\; \forall i = 1, \ldots, N \tag{B.10}$$

Suppose now that $\sum_{t=0}^{\infty} \overline{g}_i^t < \infty$. By Assumption 6 (bounded total birth and death)

$\sum_{t=0}^{\infty} b_i^t < \infty$ and $\sum_{t=0}^{\infty} d_i^t/(1-d_i^t) < \infty$. Therefore, using (B.10), we must have that $\sum_{t=0}^{\infty} |P_i^{t+1}(x) - P_i^t(x)| < \infty$, for all $x \in \mathbb{N}$. By standard results (Apostol 1974, Thm. 8.10) this implies that $\lim_{t\to\infty} P_i^t(x)$ exists for all $x \in \mathbb{N}$. This is clearly enough to prove the claim.

Q.E.D.

LEMMA B.2: *Let a vector of overall dynamics, F, be given. Suppose that $\sum_{t=0}^{\infty} \overline{g}_i^t = +\infty$. Then $\prod_{t=0}^{\infty}(1+\overline{g}_i^t)(1-n_i^{t+1}) = +\infty$*

PROOF: By standard results (Apostol 1974, Thm. 8.52), the fact that $\sum_{t=0}^{\infty} \overline{g}_i^t = +\infty$ and $\overline{g}_i^t \geq 0$ implies that $\prod_{t=0}^{\infty}(1+\overline{g}_i^t) = +\infty$. Note next that $n_i^{t+1} < 1$ for all i and t. Moreover, by bounded total birth and death (Assumption 6), $\sum_{t=0}^{\infty} |n_i^{t+1}| < \infty$. By standard results (Apostol 1974, Thm. 8.54) this implies that the product $\prod_{t=0}^{\infty}(1-n_i^{t+1})$ converges to some $k > 0$. Taking the product of the two infinite products we have just evaluated, it is now clear that $\prod_{t=1}^{\infty}(1+\overline{g}_i^t)(1-n_i^{t+1}) = +\infty$.

Q.E.D.

PROOF OF THEOREM 1: Let F be a vector of computable overall dynamics with bounded total birth and death (Assumption 6), and f be a basis for F. For all $i = 1, \ldots, N$, set $\overline{x}_i = K_i(f)$ where K_i is the total computable function of Lemma 1. Since $\overline{x}_i \in \mathcal{L}_i$, $t_i(\overline{x}_i)$ is defined and $P_i^{t_i(\overline{x}_i)}(\overline{x}_i) > 0$ for all i. By Assumption 7 (Preservation of Top Performers), $\overline{x}_i \in \mathcal{T}_i^t$ for all $t > t_i(\overline{x}_i)$. Since for all $t > t_i(\overline{x}_i)$ we also have that $\varphi_{\overline{x}_i}(h^t) \in \overline{\mathcal{A}}_i^t$, it follows that $\hat{g}_i^t = \overline{g}_i^t$ for all such values of T. Therefore, by (4)

$$P_i^t(\overline{x}_i) = P_i^{t_i(\overline{x}_i)}(\overline{x}_i) \prod_{\tau=t_i(\overline{x}_i)}^{t} (1+\overline{g}_i^\tau)(1-n_i^{\tau+1}) \quad \forall\, t > t_i(\overline{x}_i) \;\; \forall\, i = 1, \ldots, N \qquad \text{(B.11)}$$

Suppose now that $\lim_{t\to\infty} P_i^t(x)$ does not exist for some $x \in \mathbb{N}$. Then by Lemma B.2, and using (B.11), there must exist a $\overline{t} > t_i(\overline{x}_i)$ such that $P_i^{\overline{t}}(\overline{x}_i) > 1$. This contradiction is clearly enough to establish the convergence claim. The fact that the vector $\overline{x}$ can be effectively computed from f, follows automatically from the fact that we have set $\overline{x}_i = K_i(f)$ for all i.

Q.E.D.

PROOF OF REMARK 5: The claim is a trivial consequence of the Rice-Shapiro theorem (Theorem A.9), and hence we only provide a sketch of the argument. Suppose, by way of contradiction, that $\mathcal{F}$ is r.e. Then the Rice-Shapiro Theorem, directly implies that $\mathcal{F}$ must contain some f such that, say, b_1 is a Turing machine which computes a finite function (see Definition A.3). But this is clearly impossible given the definition of Computable Overall Dynamics.

Q.E.D.

PROOF OF REMARK 6: Recall that $\mathcal{F} \subset \mathbb{N}^{3N+1}$ is the set of bases for computable overall dynamics. Let $\mathcal{K}_i$ be the image of $\mathcal{F}$ under K_i so that

$$\mathcal{K}_i = \{x \in \mathbb{N} \mid \exists f \in \mathcal{F} \quad \text{such that} \quad x = K_i(f)\} \tag{B.12}$$

By way of contradiction assume now that $\mathcal{K}_i$ is r.e. By Definition A.1 this implies that for some $q \in \mathbb{N}$ we have that $x \in \mathcal{K}_i \Leftrightarrow \varphi_q(x)\downarrow$. Let now $a \in \mathbb{N}$ be such that

$$\varphi_a(f) \simeq \varphi_q(K_i(f)) \quad \forall\, f \in \mathbb{N}^{3N+1}$$

and recall that K_i is a total computable function. It follows that $\varphi_a(f)\downarrow \Leftrightarrow f \in \mathcal{F}$, which implies that $\mathcal{F}$ is r.e and hence contradicts Remark 5. This is enough to prove the claim.

Q.E.D.

PROOF OF REMARK 7: Let a vector F of computable overall dynamics be given, and let f be a basis for F. Consider now a Turing machine c_i which, given input x, performs the following computations. Compute $S(f,t)$ for successively larger values of t, beginning with $t = 0$, where S is the computable function of Remark 4. As the computation proceeds for each value of t, check whether x is born in population i, at t ($x \in \mathcal{B}_i^t$). If this is ever the case, then stop and print an output of, say, 1. If x does not ever appear in population i the computation goes on forever and hence does not halt. Clearly, by construction $\varphi_{c_i}(x)\downarrow$ if and only if $x \in \mathcal{L}_i$. By Definition A.1 this is enough to prove the claim.

Q.E.D.

PROOF OF REMARK 8: The first claim is an obvious consequence of the fact that not all Turing machines are allowable algorithms according to Definition 3 (e.g. some machines do not halt on some or all inputs). The second claim is obvious once we know that the set $\mathcal{M}_i$ of allowable algorithms is not r.e. for any i. The non recursive enumerability of each of the sets $\mathcal{M}_i$ is an easy consequence of the Rice-Shapiro theorem (Theorem A.9). We omit the details.

Q.E.D.

PROOF OF THEOREM 3: Let F be a vector of computable overall dynamics in $\Phi(\mathcal{R})$. By assumption there exists an f such that f is a basis for F and $f \in \mathcal{R}$. By (10), there exists a $v \in \mathbb{N}$ such that $f = \{\varphi_{q_1}(v), \ldots \varphi_{q_{3N+1}}(v)\}$. Using (10) again, we conclude that if $F \in \Phi(\mathcal{R})$ then there exists an f which is a basis for F and such that $K_i(f) \in \mathcal{L}_i(\mathcal{R})$, for all $i = 1, \ldots, N$. Since the support of F contains $\mathcal{L}(\mathcal{R})$, the proof of Theorem 1 applies unchanged to show that all population frequencies converge to a well defined limit. This is enough to prove the claim.

Q.E.D.

PROOF OF THEOREM 6: Let F be a vector of overall dynamics with continuous and monotonic selection dynamics, and let f be a basis for F. Since for all i the smart type $K_i(f)$ is in the support set $\mathcal{L}_i$, Theorem 1 is enough to guarantee that $\lim_{t\to\infty} P_i^t(x)$ is well defined for all $i = 1, \ldots, N$ and for all $x \in \mathbb{N}$. Let Q^t be the action frequencies associated with F, and let Q^{t_m} with $m = 0, 1, \ldots$ be a convergent subsequence of Q^t. Let also $\overline{Q} = \lim_{m\to\infty} Q^{t_m}$.

We proceed by contradiction. Suppose that the action frequencies $\overline{Q}$ are not a Nash equilibrium (pure or mixed) of Γ. Then for some $i = 1, \ldots, N$, there exists $a_i' \in \mathcal{A}_i$ and $a_i'' \in \mathcal{A}_i$ such that $\overline{Q}_i(a_i'') > 0$ (that is, action a_i'' is given positive probability in population i by the distribution $\overline{Q}$), and

$$\Pi_i(a_i', \overline{Q}_{-i}) > \Pi_i(a_i'', \overline{Q}_{-i}) \tag{B.13}$$

where the left and right-hand side of (B.13) represent the expected payoff of a_i' and a_i'' respectively, against the probability distribution $\overline{Q}_{-i}$. Using the fact that the selection

dynamics are monotonic in payoffs (see Definition 13), (B.13) implies

$$G_i\left[a_i', \overline{Q}\right] > G_i\left[a_i'', \overline{Q}\right] \tag{B.14}$$

Since $\sum_{a_i \in \mathcal{A}_i} G_i[a_i, \overline{Q}]\overline{Q}_i(a_i) = 0$, (B.14) and $Q_i(a_i'') > 0$ imply that for some $\hat{a}_i \in \mathcal{A}_i$ we have

$$G_i\left[\hat{a}_i, \overline{Q}\right] > 0 \tag{B.15}$$

Using the fact that the selection dynamics are continuous in Q (see Definition 15), (B.15) can easily be seen to imply that

$$\exists\, \overline{m}\ \ \exists\, \varepsilon > 0 \quad \text{such that} \quad G_i\left[\hat{a}_i, \overline{Q}^{t_m}\right] > \varepsilon\ \ \forall\, m \geq \overline{m} \tag{B.16}$$

Recall now that we are assuming that $K_i(f) \in \mathcal{L}_i$. It follows from (B.16) that, for $m \geq \overline{m}$, we have

$$G_i\left[\varphi_{K_i(f)}(h^{t_m}), \overline{Q}^{t_m}\right] > \varepsilon \tag{B.17}$$

But, using (4) and Assumption 6 (Bounded Total Birth and Death), (B.17) can easily be seen to contradict the fact that $\lim_{t\to\infty} P_i^t(K_i(f))$ is well defined, as required. This contradiction is enough to establish the result.

Q.E.D.

APPENDIX C

We begin this Appendix with a very brief introduction to the notion of 'provability' which we use in Section 4.3. We report it here for the sake of completeness only. We follow closely Cutland (1980, Ch. 8). Bell and Machover (1977, Ch. 7) and Mendelson (1964, Chs. 1-3), among others, provide a comprehensive treatment.

DEFINITION C.1 *(Language): Let a set of symbols $\mathcal{S}^0$ as in Theorem A.1 be given. A 'language' (based on $\mathcal{S}^0$) is the set of first order strings $\mathcal{S}^1$ derived from $\mathcal{S}^0$ as in Theorem A.1. We denote by $\Theta^1 \subset \mathcal{S}^1$, with typical element θ^1 the set of 'meaningful' first order strings. The meaningful strings of the language are simply the strings which obey some*

well defined set of syntactical rules. The elements of Θ^1 *are known as the 'formulae' or 'statements' of the language* $\mathcal{S}^1$*. We assume that the elements of* Θ^1 *can be coded into the natural numbers as in Theorem A.1. With some abuse of notation we use the same symbol for each element* θ^1 *of* Θ^1 *and its code in* $\mathbb{N}$*.*

DEFINITION C.2 *(Formal System): A Recursively Axiomatized Formal System* Λ *consists of a quadruple* $\{\mathcal{S}^1, \Theta^1, \Upsilon, \Psi\}$ *with the following meaning.* $\mathcal{S}^1$ *is a language and* Θ^1 *is the set of formulae of* $\mathcal{S}^1$ *as in Definition C.1.* Υ *is a recursive subset of* Θ^1 *(see Definition A.2) and is interpreted as the set of 'axioms' of* Λ*; any formula in* Υ *is assumed to be true in* Λ*. Finally,* Ψ *is a definition of what constitutes a 'proof' of a statement in* Θ^1 *from the axioms* Υ*. Note that a 'proof' is a finite sequence of statements in* Θ^1*. Let the set of all possible finite sequences of elements of* Θ *be denoted by* Θ^2*, with typical element* $\theta^2 = \{\theta^1_1, \ldots, \theta^1_n\}$*. Clearly (Theorem A.1), the elements of* Θ^2 *can be give a code in* $\mathbb{N}$ *in a standard way. Abusing notation again, we use the same symbol for an element* θ^2 *of* Θ^2 *and its code in* $\mathbb{N}$*.*

We require the definition Ψ *of a formal proof to have the property that given a statement* $\theta^1 \in \Theta^1$ *and a 'candidate proof'* θ^2*, it must always be possible to 'check', in a computable way, whether* θ^2 *is in fact a proof of* θ^1 *from the axioms* Υ *according to the definition* Ψ*. In other words we require that there exists a total computable function* $C : \mathbb{N}^2 \to \{0, 1\}$ *such that*

$$C(\theta^1, \theta^2) = \begin{cases} 1 & \text{if } \theta^2 \text{ is a proof of } \theta^1 \text{ from the axioms } \Upsilon \\ 0 & \text{if } \theta^2 \text{ is not a proof of } \theta^1 \text{ from the axioms } \Upsilon \end{cases} \tag{C.1}$$

Given a Recursively Axiomatized Formal System, Λ*, a statement* $\theta^1 \in \Theta^1$ *is said to be 'provable within* Λ*' if and only if there exists a finite sequence of statements* θ^2 *such that* $C(\theta^1, \theta^2) = 1$*, where* $C(\cdot, \cdot)$ *is as in (C.1).*

THEOREM C.1 *(Set of Provable Statements): In any Recursively Axiomatized Formal System the set of provable statements is recursively enumerable.*[20]

DEFINITION C.3 *(Formal Number Theory): Let* $\mathcal{S}^{1*}$ *be the language of ordinary arithmetic, with formulae* $\Theta^{1*} \subset \mathcal{S}^{1*}$*. Let* Υ^* *be the Peano axioms for ordinary arithmetic, and* Ψ^* *be the definition of proof of first order predicate calculus. Then the quadruple* $\Lambda^* = \{\mathcal{S}^{1*}, \Theta^{1*}, \Upsilon^*, \Psi^*\}$ *is referred to as Formal Number Theory.*

[20] See, for instance, Cutland (1980) Lemma 8.2.3.

THEOREM C.2 *(Axiomatized Number Theory): Formal Number Theory is a Recursively Axiomatized Formal System.*[21]

DEFINITION C.4 *(Representability): Given a Recursively Axiomatized Formal System Λ, we say that the class of statements 'f is a basis for a set of Computable Overall Dynamics' ($f \in \mathcal{F}$) is representable in Λ if and only if the following holds. For each $f \in \mathbb{N}^{3N+1}$ there exists a statement $\theta^1_f \in \Theta^1$ with the properties that: a) if θ^1_f is provable in Λ then $f \in \mathcal{F}$, b) if $\neg\theta^1_f$ ('not θ^1_f') is provable in Λ then $f \notin \mathcal{F}$, and c) θ^1_f can be obtained in a computable way from the input f.*

THEOREM C.3: *Let Λ be a Recursively Axiomatized Formal System in which the class of statements $f \in \mathcal{F}$ is representable. Then the set of $f \in \mathcal{F}$ such that the corresponding statement θ^1_f is provable within Λ is recursively enumerable.*

PROOF: Let Λ be a Recursively Axiomatized Formal System in which the class of statements $f \in \mathcal{F}$ is representable. From Definition C.4 we know that there exists a Turing machine $a \in \mathbb{N}$ such that $\varphi_a(f) = \theta^1_f$ where θ^1_f is the (code of the) counterpart of the statement $f \in \mathcal{F}$ in Λ. Since Λ is a Recursively Axiomatized Formal System, Theorem C.1 tells us that, the set of provable statements within Λ is r.e. By Definition A.1, we then know that there exists a Turing machine $b \in \mathbb{N}$ such that $\varphi_b(\theta^1) \downarrow$ if and only if θ^1 is (the code of) a provable statement in Λ. By Church's thesis and by Theorem A.3 there exists a Turing machine $d \in \mathbb{N}$ such that $\varphi_d(f) \simeq \varphi_b(\varphi_a(f))$ for all $f \in \mathbb{N}^{3N+1}$. By construction it is clear that $\varphi_d(f) \downarrow$ if and only if *both* $f \in \mathcal{F}$ and there exists a proof within Λ of the corresponding statement θ^1_f. By Definition A.1 this is enough to prove the claim.

Q.E.D.

The following is a straightforward consequence of the way we have defined the notion of Computable Overall Dynamics (see Section 3.4). However, a full proof would require a large amount of extra notation and space. For this reason we omit the details.

LEMMA C.1: *The class of statements 'f is a basis for a set of Computable Overall Dynamics' is representable within Formal Number Theory.*

An obvious consequence of Theorem C.2, C.3, and Lemma C.1 is the following.

[21] See, for instance, Mendelson (1964) Proposition 3.5.3.33 and Corollary 3.5.3.34.

LEMMA C.2: *The set of* $f \in \mathbb{N}^{3N+1}$ *such that* $f \in \mathcal{F}$ *and the corresponding statement* θ^1_f *is provable within Formal Number Theory is r.e.*

For reasons of space, we take it as given that the statement to which the next lemma refers is representable within Formal Number Theory (where representability is defined, mutatis mutandis, as in Definition C.4), and we omit a full proof.

LEMMA C.3: *Let* $\mathcal{L}(\mathcal{R}^*) = \{\mathcal{L}_1(\mathcal{R}^*), \ldots, \mathcal{L}_N(\mathcal{R}^*)\}$ *be the vector of support sets of Theorem 5. For all* i *and* $\hat{b}_i \in \mathbb{N}$ *let* $\theta^1_{\hat{b}_i}$ *be the statement of Formal Number Theory corresponding to the statement 'Turing machine* $\hat{b}_i$ *enumerates the set* $\mathcal{L}_i(\mathcal{R}^*)$ *without repetitions'. Then there exists a vector of Turing machines* $\{\hat{b}^*_1, \ldots, \hat{b}^*_N\}$ *with the following properties. For all* i, $\hat{b}^*_i$ *enumerates the set* $\mathcal{L}_i(\mathcal{R}^*)$ *without repetitions and the statement* $\theta^1_{\hat{b}^*_i}$ *is provable within Formal Number Theory.*

Lemma C.3 above is a straightforward consequence of the following facts. Firstly, both the function S of Remark 4 and the functions C_i of the proof of Lemma 1 are 'constructive' in the sense that once a complete specification of the computing devices used (Turing machines), and of the coding procedure used (cf. Theorem A.1), an actual Gödel number for Turing machines computing S and C_i for all i can be computed. Secondly, again once machines and coding are fully specified, and actual Gödel number for a Turing machine computing the function K of the *s-m-n* theorem (Theorem A.2) can be computed. Finally, by standard results (Rogers 1967, Corollary 5.2.V(c)) and by Theorem C.3, once machines and coding are fully specified again, the actual Gödel number of a Turing machine which enumerates without repetitions (cf. Theorem A.5) the set $\mathcal{R}^*$ of Theorem 5 can be computed. In other words, the Gödel numbers $\hat{b}^*_1, \ldots, \hat{b}^*_N$ can be effectively computed using a procedure which can be recognized to yield precisely such Gödel numbers. It follows that a 'proof' is then available that, for each i, Turing machine $\hat{b}^*_i$ has the required properties.

Lemmas C.4, C.5, C.6 and C.7 which follow are all direct consequences of the form of the particular statements of Formal Number Theory to which they refer. For reasons of space, we take it as given that these statements are all representable within Formal Number Theory. Full formal proofs would involve a very large amount of extra notation and space. For this reason, we omit the details.

LEMMA C.4: *For all* $b_i \in \mathbb{N}$ *let* $\theta^1_{b_i}$ *be the statement of Formal Number Theory corresponding to the statement 'Turing machine* b_i *computes a birth function which at each time*

$t = 0, 1, \ldots$ gives birth only to machine $\varphi_{\hat{b}_i}(t)$ with probability $1/(2^{t+1})$' (where $\hat{b}_i^*$ is as in Lemma C.3, for all i). Then there exists a vector of Turing machines $b^* = \{b_1^*, \ldots, b_N^*\}$ with the following properties. For each i, b_i^* computes a birth function as described, and the statement $\theta^1_{\hat{b}_i^*}$ is provable within Formal Number Theory.

LEMMA C.5: *Let θ_z^1 be the statement of Formal Number Theory (see Definition C.3) corresponding to the statement 'Turing machine z computes an information function Z such that $Z(\hat{h}^t) = t$ for all $\hat{h}^t$'. Then there exists a Turing machine z^* which computes the above information function and such that $\theta^1_{z^*}$ is provable within Formal Number Theory.*

LEMMA C.6: *Let θ_d^1 be the statement of Formal Number Theory corresponding to the statement 'Turing machine d computes the (code of the) empty set for any input n'. Then there exists a Turing machine d^* such that $\varphi_d(n)$ is the (code of the) empty set for all $n \in \mathbb{N}$, and such that the statement $\theta^1_{d^*}$ is provable within Formal Number Theory.*

LEMMA C.7: *Let $\theta^1_{g_i}$ be the statement of Formal Number Theory corresponding to the statement 'Turing machine g_i computes the function G_i of equation (11) of Definition 14 (Replicator Dynamics)'. Then for all i there exists a Turing machine g_i^* which computes the function G_i of equation (11) and such that the statement $\theta^1_{g_i^*}$ is provable within Formal Number Theory.*

PROOF OF THEOREM 5: Let F^* be the set of Computable Overall Dynamics supported by the basis $f^* = \{g_1^*, \ldots, g_N^*, b_1^*, \ldots, b_N^*, d^*, \ldots, d^*, z^*\}$, where the elements of f^* are as in Lemmas C.4, C.5, C.6 and C.7. By construction F^* satisfies all the requirements of the statement of the Theorem.

Q.E.D.

APPENDIX D

The set of computability assumptions we have stated in Section 3.4 above are more restrictive than needed for our results. In essence, they can be relaxed to deal with 'computable

real numbers' instead of restricting all variable in the model to be rational numbers. Handling real numbers in a computable framework is — to some extent at least — possible, but requires some special care.

For the sake of completeness, we outline here how our working computability assumptions of Section 3.4 can be modified to handle probabilities and growth rates which may not be rational numbers. There are many alternative ways to treat computable real-valued functions, which turn out to be broadly equivalent; we do not give any detail about this issue, but simply refer to the comprehensive work of Pour-El and Richards (1989), and to the contribution of Megiddo (1989) on computable beliefs, among others.

DEFINITION D.1 *(Computable Reals): A real number r is a computable real number if there exists a Turing machine which can compute its value up to any arbitrary degree of precision. Formally, $r \in \mathbb{R}$ is a computable real if and only if there exists a Turing machine $k \in \mathbb{N}$ such that, for any $q \in \mathbb{N}$, $|\varphi_k(q) - r| < \frac{1}{q}$.*

The notion of computable reals justifies the following alternative definition of a computable probability distribution.

DEFINITION D.2 *(Real Computable Probabilities): A probability distribution over the natural numbers, $P = \{P(0), \ldots, P(n), \ldots\}$ is said to be a real computable probability distribution if and only if there exists a Turing machine $p \in \mathbb{N}$ such that $|\varphi_p(n, q) - P(n)| < \frac{1}{q}$ for all n and q in $\mathbb{N}$.*

Notice that any probability distribution satisfying Definition 9 above, is also a real computable probability distribution according to Definition D.2. Notice further that a probability distribution over the natural which has *finite support* and such that all probabilities are computable real numbers, is automatically a real computable probability distribution according to Definition D.2. This is not necessarily true for a probability distribution over the natural numbers with an infinite support. The reason is that Definition D.2 automatically makes the support of P a *recursively enumerable* set (the proof is similar to that of Remark 7).

To extend the notion of computable selection dynamics to real-valued growth rates and real-valued probability distributions over actions, we need to extend the notion of computability to real-valued functions of real-valued vectors in $\mathbb{R}^M$. The notion of computable real vectors is an obvious extension of the notion of computable real numbers of Definition D.1. A vector $r = \{r_1, \ldots, r_M\}$ in $\mathbb{R}^M$ is a computable real vector if and only if there

exists a Turing machine $k \in \mathbb{N}$ such that, for any $q \in \mathbb{N}$, $\varphi_k(q)$ is the code of a vector $r^q = \{r_1^q, \ldots, r_M^q\}$ in $\mathbb{R}^M$ such that $|r_i^q - r_i| < \frac{1}{q}$, for all $i = 1, \ldots, M$. Informally, for a function $C : \mathbb{R}^M \to \mathbb{R}$ to be called computable we require the feasibility of *two*, related, computations. Firstly, given a computable real vector r and a desired level of approximation for $C(r)$, we must be able to compute a corresponding level of approximation for r, and secondly, given a vector which approximates r to this given level, we must be able to compute the actual value of $C(r)$ up to the desired level of approximation. Formally, we have the following.

DEFINITION D.3 *(Computable Real Function): A function $C : \mathbb{R}^M \to \mathbb{R}$ is said to be a computable real function if and only if there exist two Turing machines c_1 and c_2 with the following properties. For any $r \in \mathbb{R}^M$ which is a computable real vector, for any Turing machine k which computes approximations to r as above, and for any q in $\mathbb{N}$, $\varphi_{c_1}(q)$ computes the necessary degree of approximation for r, while $\varphi_{c_2}(\cdot)$ computes the approximate value of C (with precision $1/q$), given the approximate value of r yielded by $\varphi_{c_1}(q)$. Formally*

$$|C(r) \; - \; \varphi_{c_2}(\varphi_k(\varphi_{c_1}(q)))| < \frac{1}{q}$$

The purpose of presenting here Definitions D.1, D.2 and D.3 is to argue informally that the results of this paper are still valid when, instead of computable *rational* probabilities and computable *rational* growth rates, we assume that the selection dynamics are a set of computable real functions and all probabilities are computable real probabilities.

The key to the argument is that Remark 4 holds in an *approximate* form when the selection dynamics are a set of computable real functions and all probabilities are computable reals. The difficulties involved in this generalization are entirely related to the fact that in the construction of the 'smart machines' we described in Section 1.3, we would have to 'keep track' of several 'approximation terms' in the simulation of the overall computable dynamics. The approximation terms would have to be kept 'sufficiently small' throughout the simulation of the system forward through time, in order to ensure that the smart machine is able to take an action which is (approximately) optimal given a set of real computable overall dynamics.

REFERENCES

Abreu, D., and A. Rubinstein (1988): "The Structure of Nash Equilibrium in Repeated Games with Finite Automata," *Econometrica*, 56, 1259–81.

Anderlini, L. (1989): "Some Notes on Church's Thesis and the Theory of Games," *Theory and Decision*, 29, 19–52.

——— (1990): "Communication, Computability and Common Interest Games," Economic Theory Discussion Paper 159, Department of Applied Economics, University of Cambridge.

Anderlini, L., and H. Sabourian (1995): "Cooperation and Effective Computability," *Econometrica,* forthcoming.

Apostol, T. M. (1974): *Mathematical Analysis*. Reading, Massachusetts: Addison-Wesley.

Aumann, R. J. (1981): "Survey of Repeated Games," in *Essays in Game Theory and Mathematical Economics in Honor of Oskar Morgenstern*, pp. 11–42. Mannheim: Bibliographisches Institüt.

Aumann, R. J., and A. Brandenburger (1995): "Epistemic Conditions for Nash Equilibrium," *Econometrica*, 63, 1161–1180.

Bell, J. L., and M. Machover (1977): *A Course In Mathematical Logic*. Amsterdam: Elsevier.

Binmore, K. (1987): "Modeling Rational Players: Part I," *Economics and Philosophy*, 3, 179–214.

——— (1988): "Modeling Rational Players: Part II," *Economics and Philosophy*, 4, 9–55.

Blume, L., and D. Easley (1992): "Evolution and Market Behavior," *Journal of Economic Theory*, 58, 9–40.

Brown, G. (1951): "Iterated Solution of Games by Fictitious Play," in *Activity Analysis of Production and Allocation*, ed. by T. C. Koopmans, no. 13 in Cowles Commisision for Research in Economics Monographs, pp. 374–376. New York: Wiley.

Canning, D. (1992a): "Average Behavior in Learning Models," *Journal of Economic Theory*, 57, 442–72.

——— (1992b): "Rationality, Computability, and Nash Equilibrium," *Econometrica*, 60, 877–88.

Cutland, N. J. (1980): *Computability: An Introduction to Recursive Function Theory*. Cambridge: Cambridge University Press.

Davis, M. (1958): *Computability and Unsolvability*. New York: Dover Publications.

Foster, D., and H. P. Young (1991): "Cooperation in the Short and in the Long Run," *Games and Economic Behavior*, 3, 145–156.

Fudenberg, D., and D. M. Kreps (1991): "A Theory of Learning, Experimentation and Equilibrium in Games," Stanford University, mimeo.

Fudenberg, D., and D. K. Levine (1993a): "Self Confirming Equilibrium," *Econometrica*, 61, 523–45.

——— (1993b): "Steady State Learning and Nash Equilibrium," *Econometrica*, 61, 547–73.

Hofbauer, J., and K. Sigmund (1988): *The Theory of Evolution and Dynamical Systems*. Cambridge: Cambridge University Press.

Howard, J. V. (1988): "Cooperation in the Prisoners' Dilemma," *Theory and Decision*, 24, 203–213.

Jordan, J. S. (1991): "Bayesian Learning in Normal Form Games," *Games and Economic Behavior*, 3, 60–81.

Kalai, E., and E. Lehrer (1993): "Rational Learning Leads to Nash Equilibrium," *Econometrica*, 61, 1019–45.

Kandori, M., G. J. Mailath, and R. Rob (1993): "Learning, Mutation, and Long Run Equilibria in Games," *Econometrica*, 61, 29–56.

Marimon, R., and E. McGrattan (1992): "On Adaptive Learning in Strategic Games," University of Minnesota, mimeo.

McAfee, R. P. (1984): "Effective Computability in Economic Decisions," University of Western Ontario, mimeo.

Megiddo, N. (1986): "Remarks on Bounded Rationality," Research Report RJ5270, IBM Almaden Research Center, San Jose, California.

——— (1989): "On Computable Beliefs of Rational Machines," *Games and Economic Behavior*, 1, 144–69.

Megiddo, N., and R. Widgerson (1987): "Turing Machines Play the Finitely Repeated Prisoners' Dilemma," IBM Almaden Research Center, San Jose California, mimeo.

Mendelson, E. (1964): *Introduction to Mathematical Logic*. Princeton: D. Van Nostrand.

Milgrom, P. R., and J. Roberts (1990): "Rationalizability, Learning, and Equilibrium in Games with Strategic Complementarities," *Econometrica*, 58, 1255–77.

——— (1991): "Adaptive and Sophisticated Learning in Normal Form Games," *Games and Economic Behavior*, 3, 82–100.

Nachbar, J. H. (1990): "'Evolutionary' Selection Dynamics in Games: Convergence and Limit Properties," *International Journal of Game Theory*, 19, 59–89.

Neyman, A. (1985): "Bounded Complexity Justifies Cooperation in the Finitely Repeated Prisoners' Dilemma," *Economics Letters*, 19, 227–229.

Pour-El, M., and J. Richards (1989): *Computability in Analysis and Physics*. Berlin: Springer-Verlag.

Rogers, H. (1967): *Theory of Recursive Functions and Effective Computability*. London: McGraw-Hill Book Company.

Rubinstein, A. (1986): "Finite Automata Play the Repeated Prisoner's Dilemma," *Journal of Economic Theory*, 39, 83–96.

Samuelson, L., and J. Zhang (1992): "Evolutionary Stability in Asymmetric Games," *Journal of Economic Theory*, 57, 363–91.

Selten, R. (1991): "Evolution, Learning, and Economic Behavior," *Games and Economic Behavior*, 3, 3–24.

Shapley, L. S. (1964): "Some Topics in Two-Person Games," in *Advances in Game Theory*, ed. by M. Dresher, L. S. Shapley,and W. Tucker, no. 52 in Annals of Mathematics Studies, pp. 1–28. Princeton: Princeton University Press.

Spear, S. E. (1989): "Learning Rational Expectations under Computability Constraints," *Econometrica*, 57, 889–910.

VanDamme, E. (1987): *Stability and Perfection of Nash Equilibria*. Berlin: Springer-Verlag.

Weibull, J. W. (1992): "An Introduction to Evolutionary Game Theory," Working Paper 347–1992, Industriens Utredningsinstitut, Stockholm.

Young, H. P. (1993): "The Evolution of Conventions," *Econometrica*, 61, 57–84.