Mathematical specification of an agend-based model of exchange

Nicola Botta[1] and **Antoine Mandel**[2] and **Mareen Hofmann**[3] and **Sibylle Schupp**[4] and **Cezar Ionescu**[5]

1 Introduction

Agent-based computer models are becoming increasingly popular in economics, both as a toolbox for theoreticians [12] and for policy advise [7].

All science nowadays relies on computer models. For an interesting analysis of just how much, see "Simulation and its Discontents" [13]. But a distinguishing feature is that, in computational economics, computer models are often the only kind of models we have.

The situation is completely different from physics and engineering where computer models are secondary to mathematical ones. It is the latter which embody the theories and serve as object of discussion between scientists: computer code rarely makes its way in scientific publications.

In physics and engineering a computer model is judged to be correct if it faithfully implements its mathematical counterpart. Prominent computer model failures – from 1996 Ariane 5 flight 501 [1] to 2010 "flash crash" at the New York Stock Exchange – remind us that, sometimes, it is difficult to formulate exactly what counts as "faithful". But, at least in principle, mathematical models provide a "golden standard" for the correctness of computer models in physics and engineering.

But what is "golden standard" against which to judge the correctness of agent-based models for computational economics ? When can we trust the outcome of multi-agent simulations designed to study viable decarbonization options or to inform politicians on the effect of green building subsidies on unemployment ?

Such models are usually accompanied by a narrative, an informal description of the ideas behind their development. But a narrative is too blunt an instrument to help us decide whether a computer implementation of it is correct or not, whether the results of a simulation are trustworthy or flawed by programming errors.

In a nutshell, the current state of affairs in computational economics can be summed up as follows: we can limit ourselves to informal narratives, or we can use simulations of computer models which we do not understand [10] and whose correctness we cannot guarantee.

[1] Potsdam Institute for Climate Impact Research
[2] Centre d'économie de la Sorbonne, Université Paris 1
[3] Freie Universität Berlin
[4] Technische Univertität Hamburg-Harburg
[5] Potsdam Institute for Climate Impact Research

In agent-based simulations of financial markets, the deployment of proprietary toolkits and platforms makes this situation even worse.

1.1 Gintis' model

In 2006, Herbert Gintis [5] announced the discovery of a mechanism that would explain price formation and disequlibrium adjustment without requiring the presence of a central authority as is currently assumed in mainstream economics.

Gintis' results were, as he put it "empirical rather than theoretical: we have created a class of economies and investigated their properties for a range of parameters." They were obtained by computer simulations.

The results were relevant – understanding price formation is a long-standing puzzle in economic theory – and intriguing: which class of economies is actually specified by the model presented in [5] ? Can one express the properties investigated in [5] formally ?

In 2009 two groups of researchers, one at PIK, the other at Chalmers [4], independently attempted to do something which should perhaps be routine, but is hardly ever done: to reimplement the model and reproduce the results reported in [5].

After initial attempts failed and Gintis graciously provided the source code, both groups discovered several ways that his implementation diverged from the description in the paper, one of which could be called a "bug". Much more problematic, however, was the ambiguity left open by the narrative given in [5]. This is quite typical in computational economy and econophysics: scientists tend to believe that the mathematical equations and the narrative used to describe a model are sufficient specifications for the implementation of that model but this is rarely the case.

1.2 Our contribution

In [3] we proposed a functional framework for the specifications of computer models of exchange. The framework provides, among others, the notions of bilateral exchanges, bilateral trades, trading policies and games and dynamical models of exchange.

There, we addressed the question of how to describe and specify computer-based dynamical models of exchange in a language which is more accessible to non-programmers than

program listings and yet less ambiguous than narrative descriptions.

In this contribution, we apply that framework and derive a complete mathematical specification of Gintis' model [5][6].

We first derive a bona-fide specification of the model on the basis of [5] and of the model implementation kindly made available by the author. From this specification, we derive a first model reimplementation. This fails to reproduce the results reported in [5]. This is not really surprising: in the original model, exchanges between agents are rationed. In contrast, the results presented in [8] suggest that convergence of agent-specific prices towards the "special" equilibrium prices discovered in [5] depend on a "no strategic rationing condition".

Then, we show how the model specification can be relaxed for the corresponding implementation to reproduce the original results. The relaxed specification is consistent with the results reported in [8]. It suggests that trade resolving policies that yield convergence of agent-specific prices (towards equilibrium prices at large times) cannot grant convergence of allocations (towards equilibrium allocations, at fixed equilibrium prices).

2 Notation

In this section we introduce the basic notation used throughout this paper. The presentation is intentionally terse, details and motivations are discussed in [3].

We use A and G to denote finite sets of agents and goods.

Stocks are formulated as functions of type $G \to \mathbb{R}_{\geq 0}$. Prices are formulated as functions of type $G \to \mathbb{R}_{>0}$ and utilities are formulated as functions of type $(G \to \mathbb{R}_{\geq 0}) \to \mathbb{R}$. For convenience, we introduce the abbreviations $Q = G \to \mathbb{R}_{\geq 0}$, $P = G \to \mathbb{R}_{>0}$ and $U = Q \to \mathbb{R}$. Allocations and utility profiles are formulated as functions of type $A \to Q$ and $A \to U$, respectively.

We write $q : Q$ to posit that q is a stock that is, a function of type $G \to \mathbb{R}_{\geq 0}$. We denote function application by juxtaposition: if $x : A \to Q$, $x\ a : Q$ is the stock of $a : A$ according to x and $x\ a\ g : \mathbb{R}_{\geq 0}$ the quantity of $g : G$ according to $x\ a$.

This notation is standard in mathematics and computing science but not quite common in engineering and computational economics[7]. For a discussion of the advantages and disadvantaged of the two notations we refer the reader to [3].

In the next section we apply the framework presented in [3] to derive a complete mathematical specification of the model presented in [5]. Terms that denote notions specified in [3] are introduced in italics and the relevant sections of [3] are given in footnotes, e.g., *bilateral exchange*[8].

3 A mathematical specification of [5]

In a nutshell, the model presented in [5] is a *dynamical model of exchange*[9] for so-called private prices. These are agent-specific prices. At each iteration step, the agent-specific prices are updated according to an evolutionary algorithm. This is driven by a trading fitness. The trading fitness is measured in a *trading game*[9] which, in turn, is controlled by the actual prices. Strictly speaking, the trading game is not a model of exchange: in [5], agents are not only trading but also producing and consuming.

We formulate time-dependent agent-specific prices as a function $y : \mathbb{N} \to A \to P$. In the model, $y\ 0$ is given and $y\ 1, y\ 2, \ldots$ are computed by iterating an evolutionary algorithm [11]. This is obtained by folding a copy-mutate rule *cm* on a copy-mutate schedule:

$$\begin{aligned} y\ 0 &= y_0 \\ y\ (t+1) &= \text{fold}\ \ cm\ (y\ t)\ (cms\ t) \end{aligned} \tag{1}$$

In the above specification, $cms\ t$ is a random list of pairs

$$cms\ t\ :\ List\ ((A \times A) \times (G \to [0,1]))\ .$$

According to the signature of fold[10], *cm* maps agent-specific prices and pairs of type $(A \times A) \times (G \to [0,1])$ into agent-specific prices:

$$cm\ :\ (A \to P) \to (A \times A) \times (G \to [0,1]) \to (A \to P)$$

In the rest of this section, we refine (1) by putting forward specifications for *cm* and *cms*. We start by formulating the model setup. Then, we specify *cm* in terms of a trading fitness function $f : A \to \mathbb{R}$ and explain how random copy-mutate schedules are drawn. Finally, we specify the trading game which defines f. This completes the specification of the model presented in [5].

3.1 Model setup

The copy-mutate rule *cm* and the random schedules $cms\ t, t \in \mathbb{N}$ depend on a number of functions and parameters. These are:

- a *sector*[11] function s. Its specification (equation 20 of [3]) reads

$$\begin{aligned} &s : A \to G \\ &\forall\ g \in G\ \ s^{-1}\ g \neq \emptyset\ . \end{aligned}$$

- An initial allocation $x_0 : A \to Q$.
- A utility profile $u : A \to U$.
- A sector-to-sector *number of peers*[11] np. Its specification (equation (23) of [3]) reads

$$\begin{aligned} &np : G \to G \to \mathbb{N} \\ &\forall\ g, g' \in G\ \ np\ g\ g' \leq |s^{-1}\ g'|\ . \end{aligned}$$

[6] By complete specifications we mean specifications that describe the model entirely and from which model implementations can be derived without ambiguity.

[7] In economics, stocks, for instance, are often formulated in terms of vectors in $\mathbb{R}^n_{\geq 0}$ where $n = |G|$ and one writes $q \in \mathbb{R}^n_{\geq 0}$ and q_j instead of $q : G \to \mathbb{R}_{\geq 0}$ and $q\ g$

[8] Section 3.3 of [3].

[9] Section 3.6 of [3].

[10] fold is a polymorphic function. Its type depends on two parameters: $\text{fold} : (X \to Y \to X) \to X \to List\ Y \to X$. Fold reduces lists of arbitrary type Y to a single value of (another arbitrary) type X: it takes a binary function, an initial value (the accumulator), and a list. It first calls this binary function with the initial value and the rightmost element of the list, to obtain a new accumulator, and then repeats for the remaining elements; in the end, it returns one single value, e.g., $\text{fold}\ +\ 0\ [3,2,1] = (((0+1)+2)+3) = 6$. For a definition of fold see [3] or standard computing science textbooks.

[11] Section 3.5 of [3].

- A copy-mutate fraction $cmf \in [0,1] \subset \mathbb{R}$.
- A mutation probability $mp \in [0,1] \subset \mathbb{R}$.
- A mutation factor $mf \in (0,1) \subset \mathbb{R}$.
- A number of *trading rounds*[12] $n_r \in \mathbb{N}$.

The model is based on a number of specific assumptions. These are:

(a) The number of goods is greater than one: $|G| > 1$.

(b) Sectors are not empty and equally populated. The number of agents per sector is denoted by $n_a \in \mathbb{N}$:

$$\forall\ g \in G\ \ |s^{-1}\ g| = |A|/|G| = n_a > 0$$

(c) The sector-to-sector number of peers is constant. The number of peers is denoted by $n_p \in \mathbb{N}$:

$$\forall\ g, g' \in G\ \ np\ g\ g' = n_p$$

(d) The initial allocation x_0 is sector-wise constant. Moreover, in any sector, x_0 is different from zero only in the sector-specific good:

$$\forall\ a, a' \in A\ \ s\ a = s\ a'\ \Rightarrow\ x_0\ a = x_0\ a'$$
$$x_0\ a\ g \neq 0\ \Rightarrow\ g = s\ a\ .$$

(e) The utility profile is constant. The utility function of each agent is the Scarf utility function[13] with $\lambda = 1/|A|$:

$$\forall\ a \in A\ \ u\ a\ y = \min_{g \in G}\ (y\ g)\ /\ (w\ g) \tag{2}$$

$$\text{where}: w = \frac{1}{|A|} * \sum_{a \in A} x_0\ a \tag{3}$$

3.2 The copy-mutate rule

In the prices iteration (1), the copy-mutate rule *cm* instantiates a replicator dynamic [11]:

$$z' = cm\ z\ ((a_1, a_2), \xi)$$
$$\Rightarrow$$
$$z'\ a \neq z\ a\ \Rightarrow\ a = a_1 \vee a = a_2$$
$$\wedge$$
$$f\ a_1 < f\ a_2$$
$$\Rightarrow$$
$$z'\ a_2 = z\ a_2\ \wedge$$
$$z'\ a_1\ g = \begin{cases} z\ a_2\ g & \text{if } \xi\ g < mp \\ (z\ a_2\ g)/mf & \text{if } \xi\ g \geq mp \wedge \xi\ g < 0.5 \\ (z\ a_2\ g) * mf & \text{if } \xi\ g \geq mp \wedge \xi\ g \geq 0.5 \end{cases}$$
$$\wedge$$
$$f\ a_1 \geq f\ a_2$$
$$\Rightarrow$$
$$z'\ a_1 = z\ a_1\ \wedge$$
$$z'\ a_2\ g = \begin{cases} z\ a_1\ g & \text{if } \xi\ g < mp \\ (z\ a_1\ g)/mf & \text{if } \xi\ g \geq mp \wedge \xi\ g < 0.5 \\ (z\ a_1\ g) * mf & \text{if } \xi\ g \geq mp \wedge \xi\ g \geq 0.5 \end{cases}$$

[12] Section 3.5 of [3].
[13] Section 3.1 of [3].

As mentioned above, *cm* depends on a trading fitness f. Thus, the specification of *cm* is, strictly speaking, incorrect. While *mf*, *mp* are constant values provided by the model setup given in section 3.1, f is an undefined function of the system's *state*[9]. We specify the computation of f in section 3.4.

3.3 The copy-mutate schedule

Each copy-mutate schedule *cms* t is defined in terms of two random functions. The first one is an inverse numbering[14] of G

$$\gamma\ :\ [\,0, |G|\,) \to G\ .$$

There are $|G|!$ such numberings. The second random function provides, for each sector $g \in G$, $cmf * n_a$ random pairs[15]. The first element of each pair is itself a pair of agents in $s^{-1}\ g$. The second element are $|G|$ random numbers in $[0,1] \subset \mathbb{R}$

$$\beta\ :\ G \to [\,0, cmf * n_a\,) \to (A \times A) \times (G \to [0,1])\ .$$

Thus, $\beta\ g\ k$ is a partial function and fulfills the specification $\forall\ g \in G, \forall\ k \in [\,0, cmf * n_a\,)$[16]

$$\text{ran}\,(\beta\ g\ k)\ =\ \big((s^{-1}\ g) \times (s^{-1}\ g)\big) \times (G \to [0,1])\ .$$

Each draw is assumed to be equally probable. Given a draw (γ, β), the schedule is defined by the list comprehension[17]

$$[\ \beta\ (\gamma\ j)\ k\ |\ j \leftarrow [\,0, |G|\,),\ k \leftarrow [\,0, cmf * n_a\,)\ ]$$

Thus, the schedule is the list of pairs $\beta\ (\gamma\ j)\ k$ obtained by drawing j and k from $[\,0, |G|\,)$ and $[\,0, cmf * n_a\,)$, respectively. It consists of $|G| * (cmf * n_a)$ elements.

3.4 The trading fitness

The trading fitness $f : A \to \mathbb{R}$ is computed in a trading game of the kind formulated in section 3.6 of [3]:

$$f = h\ (\text{map}\ (\text{fold}\ (cp.tr)\ (f_0, x_0)\ )\ tss)\ . \tag{4}$$

In the above expression, $\text{map} : (X \to Y) \to List\ X \to List\ Y$ is a function that transforms lists: it takes a unary function and applies it element-wise to the list. For instance, map negate $[1,2,3] = [-1,-2,-3]$. The function $cp.tr$ represents the composition of *cp* and *tr*: $(cp.tr)\ a\ b = cp\ (tr\ a\ b)$. The target of h is equal to $A \to \mathbb{R}$. The game consists of n_r rounds. The elements of *tss* are random trading schedules,

[14] a numbering of a finite set X is a bijective function of type $X \to [\,0, |X|\,)$. It associates to each element of X exactly one number between 0 and $|X| - 1$. There are $|X|!$ such functions.
[15] For sake of simplicity, we just write $cmf * n_a$ to denote the natural number obtained by rounding the product between $cmf \in \mathbb{R}$ and $n_a \in \mathbb{N}$.
[16] we could make $\beta\ g\ k$ total by introducing dependent types. With dependent types, one could express the specification for $\beta\ g\ k$ through the type system. Applying dependent types for specifications is a natural approach but goes beyond the scope of this paper.
[17] Informally, comprehension on lists is analog to set comprehension. For instance $\{2 * i\ |\ i \in \{0, \ldots, n\}\}$ translates to $[2 * i\ |\ i \leftarrow [0..n]]$ on lists. This constructs the list of integers $[0, 2, \ldots, 2 * n]$ from $[0..n]$. The latter denotes the list of integers $0, 1, \ldots$ until n. The expression $a \leftarrow as$ is read "for a drawn from as". It represents the action of iterating over the elements of as in the order defined by as.

one schedule per round. The outcome of a round is a fitness-allocation pair:

$$h : List\ ((A \to \mathbb{R}) \times (A \to Q)) \to (A \to \mathbb{R})\ . \tag{5}$$

In every round, the initial fitness f_0 is zero for all agents:

$$\forall\ a \in A\ \ f_0\ a = 0\ .$$

The initial allocation x_0 (and the number of rounds n_r) are given by the model setup, see section 3.1. We specify the computation of f by giving h, cp, tr and by describing how the random schedules of tss are drawn.

3.4.1 The random schedules

Each schedule in tss is defined in terms of three random functions. The first one is an inverse numbering of G like the one introduced above for the copy-mutate schedules. For the sake of simplicity we call this function γ, too. There are $n_\gamma = |G|!$ such numberings. The second function provides $|G|$ inverse numbering, one for each sector

$$\alpha\ :\ G \to [\,0, n_a\,) \to A\ .$$

The idea is that $\alpha\ g$ is an inverse numbering of $s^{-1}\ g$. Therefore $\alpha\ g$ fulfills

$$\forall\ g \in G\ \ \mathrm{ran}\,(\alpha\ g)\ =\ s^{-1}\ g \tag{6}$$

There are $n_\alpha = (n_a!)^{|G|}$ such numberings. The third function gives, for each pair of sectors g, $g' \neq g$ and for each agent in $s^{-1}\ g$, a random draw of n_p agents in $s^{-1}\ g'$

$$\eta\ :\ G \to G \to A \to \mathbb{N} \to A\ .$$

Thus, $\eta\ g\ g'$ is a partial function and fulfills the specification[18]

$$\forall\ g, g' \in G\ \ g' \neq g\ \Rightarrow\ \mathrm{dom}\,(\eta\ g\ g')\ =\ s^{-1}\ g\ , \tag{7}$$

$$\forall\ g, g' \in G\ \ g' \neq g\ \Rightarrow\ \mathrm{ran}\,(\eta\ g\ g')\ =\ [\,0, n_a\,) \to s^{-1}\ g'\ . \tag{8}$$

There are $n_\eta = (n_a!/(n_a - n_p)!)^{n_\alpha * (|G|-1)^{|G|}}$ such random draws. Thus, at each round, there are $n_\gamma^{n_\alpha * n_\eta}$ ways to define a random trading schedule, each one represented by a draw (γ, α, η) satisfying (6)-(8). Each draw is assumed to be equally probable. Given a draw (γ, α, η), the schedule is defined by the list comprehension

$$\begin{aligned}[\big(\alpha\ (\gamma\ j)\ i, \eta\ (\gamma\ j)\ (\gamma\ j')\ (\alpha\ (\gamma\ j)\ i)\ k\big)\ \mid\ & j \leftarrow [\,0, |G|\,),\\ & j' \leftarrow [\,0, |G|\,),\\ & j' \neq j,\\ & i \leftarrow [\,0, n_a\,),\\ & k \leftarrow [\,0, n_p\,)\ ]\end{aligned}$$

[18] As for the random function β introduced for the specification of the copy-mutate schedules, we could make $\eta\ g\ g'$ total by introducing dependent types.

3.4.2 The trade function tr and the function cp

As mentioned above, the trading game (4) is not a model of exchange in the sense made clear in [3]: after a trade, the two agents may undergo a consumption-production step. This is represented by the function cp. This function is composed with an *extended elementary bilateral trade* tr[9]. We start with the specification of tr:

$$\begin{aligned}tr : (A \to \mathbb{R}) \times (A \to Q) \to A \times A \to\\ ((A \to \mathbb{R}) \times (A \to Q)) \times (A \times A)\end{aligned} \tag{9}$$

$$tr\ (f, x)\ (a_1, a_2) = ((f', x'), (a_1', a_2'))$$

$$\Rightarrow$$

$$f' = f\ \wedge\ a_1' = a_1\ \wedge\ a_2' = a_2 \tag{10}$$

$$x(\mathcal{X}_e\ a_1\ a_2\ g_2\ g_1)x' \tag{11}$$

$$(x'\ a_1 - x\ a_1)\ g_2 = \delta_1 \tag{12}$$

$$(x'\ a_2 - x\ a_2)\ g_1 = \delta_2 \tag{13}$$

$$(\delta_1, \delta_2) = trp\ p_1\ p_2\ (o_1, g_1)\ (d_1, g_2)\ (o_2, g_2)\ (d_2, g_1) \tag{14}$$

$$(o_1, d_1) = odp\ a_1\ p_1\ g_2 \tag{15}$$

$$(o_2, d_2) = odp\ a_2\ p_2\ g_1 \tag{16}$$

$$g_1 = s\ a_1 \tag{17}$$

$$g_2 = s\ a_2 \tag{18}$$

$$p_1 = y\ t\ a_1 \tag{19}$$

$$p_2 = y\ t\ a_2 \tag{20}$$

The types of the two arguments taken by tr, a fitness-allocation pair and a pair of agents, is dictated by equations (4) and (5). The result of applying tr to (f, x), (a_1, a_2) is a pair $((f, x'), (a_1, a_2))$, see (10). The new allocation, x', is related to x through an *elementary bilateral exchange* of goods $s\ a_2$ and $s\ a_1$. The specification of elementary bilateral exchanges is given in section 3.3 of [3]. One could replace equation (11) with specifications (11), (12) of [3][19] with $g_1 = s\ a_2$ and $g_2 = s\ a_1$.

Equations (12)-(20) are refinements of (11). They completely define the outcome of the trade x' in terms of the *offer and demand policies*[20] of a_1, a_2 – the function odp – and of an agent-independent trade-resolving policy trp. These functions depend on the actual allocation x, on the model setup functions u, x_0, s and on the actual prices $y\ t$ of the iteration in which the trading game is played. As we will see, trp and, therefore, the elementary bilateral trade tr are not symmetric. In general,

$$tr\ (f, x)\ (a_1, a_2) \neq tr\ (f, x)\ (a_2, a_1)\ .$$

In [5], the first agents in the pairs of a trading schedule are called demanders. The second agents are called responders. In the following, we will use the same terminology.

The offer and demand policy is *demand-based* and *value-preserving*[13]. With u, x_0 as in the model setup of section 3.1 and according to equation (8) of [3], the demand of a under

[19] Or, equivalently, with (9), (10) and (12) of [3] with $g_1 = s\ a_2$ and $g_2 = s\ a_1$.

[20] Section 3.4 of [3].

the agent-specific prices of the t-th iteration $y\ t$ is

$$\frac{(x_0\ a)\cdot(y\ t\ a)}{(\sum\limits_{a\in A} x_0\ a)\cdot(y\ t\ a)} * (\sum_{a\in A} x_0\ a)\ .$$

Thus, for fixed x_0, u (model setup) and time dependent prices $y\ t$ (we are considering a trading game at fixed prices), the excess demand profile for $x : A \to Q$ is

$$\begin{aligned} &ed\ x\ : A \to Q \\ &ed\ x\ a = \frac{(x_0\ a)\cdot(y\ t\ a)}{(\sum\limits_{a\in A} x_0\ a)\cdot(y\ t\ a)} * (\sum_{a\in A} x_0\ a)\ -\ (x\ a) \end{aligned}$$

The offer and demand policy is defined in terms of such excess demand

$$\begin{aligned} &odp : A \to P \to G \to \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \\ &g \neq (s\ a)\ \wedge\ odp\ a\ p\ g = (o, d)\ \Rightarrow \\ &d\ =\ \begin{cases} 0 & \text{if } ed\ x\ a\ g \leq 0 \\ ed\ x\ a\ g & \text{if } ed\ x\ a\ g > 0 \\ & \quad \wedge \\ & \quad (ed\ x\ a\ g) * \dfrac{p\ g}{p\ (s\ a)} \leq x\ a\ (s\ a) \\ (x\ a\ (s\ a)) * \frac{p\ (s\ a)}{p\ g} & \text{if } ed\ x\ a\ g > 0 \\ & \quad \wedge \\ & \quad (ed\ x\ a\ g) * \dfrac{p\ g}{p\ (s\ a)} > x\ a\ (s\ a) \end{cases} \\ &o\ =\ d * \frac{p\ g}{p\ (s\ a)} \end{aligned}$$

and by requiring the value of the offer to equal the value of the demand. Notice that $odp\ a\ g$ is only defined for $g \neq (s\ a)$[21] and fulfills specification (13) of [3].

The description of the trading mechanism given in [5] does not yield an unambiguous specification of *trp*. However, the author has provided a complete implementation of the model presented in [5] in Delphi. This is an extension of Object Pascal which provides primitives for implementing applications based on graphical user interfaces on Microsoft Windows operating systems. The implementation is consistent with the following specification:

$$\begin{aligned} &trp : P \to P \to \mathbb{R}_{\geq 0} \times G \to \mathbb{R}_{\geq 0} \times G \to \mathbb{R}_{\geq 0} \times G \to \\ &\qquad \mathbb{R}_{\geq 0} \times G \to \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \\ &g_1 \neq g_2\ \wedge\ trp\ p_1\ p_2\ (o_1, g_1)\ (d_1, g_2)\ (o_2, g_2)\ (d_2, g_1) = (\delta_1, \delta_2) \\ &\quad \Rightarrow \\ &(\delta_1, \delta_2)\ =\ \begin{cases} (0,0) & \text{if } o_1 * (p_2\ g_1) < d_1 * (p_2\ g_2) \\ (\delta_1', \delta_2') & \text{otherwise} \end{cases} \qquad (21) \\ &(\delta_1', \delta_2')\ =\ \begin{cases} (d_2 * \frac{p_1\ g_1}{p_1\ g_2}, d_2) & \text{if } d_2 < \delta_2'' \\ (\delta_1'', \delta_2'') & \text{otherwise} \end{cases} \qquad (22) \\ &(\delta_1'', \delta_2'')\ =\ \begin{cases} (o_2, o_2 * \frac{p_1\ g_2}{p_1\ g_1}) & \text{if } o_2 < \delta_1''' \\ (\delta_1''', \delta_2''') & \text{otherwise} \end{cases} \qquad (23) \\ &(\delta_1''', \delta_2''')\ =\ \begin{cases} (x_2\ g_2, x_2\ g_2 * \frac{p_1\ g_2}{p_1\ g_1}) & \text{if } x_2\ g_2 < d_1 \\ (d_1, o_1) & \text{otherwise} \end{cases} \qquad (24) \end{aligned}$$

[21] as discussed for the random functions which define the trading schedules, we could make *odp* total by introducing dependent types.

In section 4 we provide some experimental support for this claim. As anticipated, *trp* is not symmetric: if the value of the demander's offer meets or exceeds the value of its demand (according to the responder's prices), the outcome of the trade is liable to be the demander's offer and demand, see equation (21). For this to be the case, three further conditions have to be met.

The first one, equation (24), is a budget condition. It requires the amount of g_2 to be exchanged not to exceed the stock of the offerer. If this happens, the demander's demand is capped to the responder's stock. This ensures that, after exchange, the responder's stock of g_2 does not turn negative.

The second and the third are rationing constraints. They require the amount of g_2 exchanged not to exceed the offer (of g_2) of the responder, equation (23) and the amount of g_1 to be exchanged not to exceed the demand of the responder, equation equation (22). We will discuss the effects of these conditions on the dynamics of prices in section 4.

Notice that, with *trp* defined as above, the outcome of a trade does not, in general, fulfill specification (14) of [3]. On the other hand, *odp*, *trp* ensure that *tr* is value-preserving: if $p_1 = p_2$, the exchange of δ_1 and δ_2 does not modify the value of the stocks of both agents.

Let's now turn our attention to the consume-produce function *cp*. The type of *cp* can be inferred from the type of *tr* (9) and from equation (4):

$$cp : ((A \to \mathbb{R}) \times (A \to Q)) \times (A \times A) \to (A \to \mathbb{R}) \times (A \to Q)$$

The consume-produce function *cp* takes the outcome of *tr* — a fitness-allocation pair and a pair of agents — and returns a new fitness allocation pair. The rationale of *cp* is twofold.

On one hand, it accounts for successful trades — trades which have modified the stocks of the interacting agents — in the new trading fitness. Remember that all agents compute their offer and demand according to the same policy *odp*. In other words, two agents with equal initial and actual stocks and with the same (agent-specific) prices issue equal offers and demands. Thus, since initial allocations are sector-wise constant, the trading fitness is, within a given sector, a measure of the fitness of the agent-specific prices. This explains why prices, in [5] are sometimes referred to as *strategies*.

On the other hand, *cp* rewards successful agents by providing them with even more goods to trade (production). From the listing made available by Gintis, we deduce the specification:

$$\begin{aligned} &cp\ ((f, x), (a_1, a_2)) = (f', x') \\ &\quad \Rightarrow \\ &f'\ a \neq f\ a \Rightarrow a = a_1 \vee a = a_2 \\ &x'\ a \neq x\ a \Rightarrow a = a_1 \vee a = a_2 \\ &a \in \{a_1, a_2\} \Rightarrow f'\ a = f\ a + u\ a\ (x\ a) \qquad (25) \\ &a \in \{a_1, a_2\} \Rightarrow \\ &\quad x'\ a = \begin{cases} x_0\ a & \text{if } u\ a\ (x\ a) \geq 1 \\ (x\ a) * (1 - u\ a\ (x\ a)) & \text{otherwise} \end{cases} \qquad (26) \end{aligned}$$

This specification is puzzling in a number of ways. The new stocks — or, from a modeling perspective, production processes — depend on the utility in a very discontinuous fashion. Agents which have achieved levels of utility near but below one are forced to reduce their stocks — to consume —

proportionally to the achieved utilities. No production takes place. With little stocks left to trade, such agents are not likely to improve their utilities in later interactions. Their stocks will be further reduced albeit at increasingly slower rates. On the other hand, the stocks of agents which have managed to achieve utilities equal or greater than one are set back to their initial stocks. This resetting can be interpreted as a consumption-production step. After such step, agents enter the next trade with zero utility and full budget. It is not obvious (to us) what is the motivation behind the utility threshold and why such threshold is set to one. Notice, however, that, with x_0, u and λ as defined in the model setup 3.1, one has

$$(x,p)\mathcal{E}(x_0,u) \;\Rightarrow\; u\ a\ (x\ a) = \frac{(x'_0\ (s\ a)) * (p\ (s\ a))}{\frac{1}{|G|} * \sum_{g' \in G} (x'_0\ g') * (p\ g')} \quad (27)$$

where $x'_0 : G \to \mathbb{R}_{\geq 0}$ is the function that defines a sector-wise constant initial allocation x_0 which complies with assumption 3 from section 3.1:

$$x_0\ a\ g = \begin{cases} x'_0\ g & \text{if } g = s\ a \\ 0 & \text{otherwise .} \end{cases} \quad (28)$$

The numerical results presented in [5] suggest that, at large times, the prices

$$\begin{aligned} &\hat{p} : P \\ &\hat{p}\ g = \frac{1}{w\ g} = \frac{|A|}{\sum_{a \in A} (x_0\ a\ g)} = \frac{|G|}{x'_0\ g} \end{aligned} \quad (29)$$

are, under (1), more stable than any other prices. Specifically, the author has observed that, independently of the initial prices y_0, the price iteration tends to "converge", at long times, towards the "special" prices (29)[22]. For $p = \hat{p}$ equation (27) becomes

$$(x,p)\mathcal{E}(x_0,u) \;\Rightarrow\; u\ a\ (x\ a) = 1$$

Thus, in [5], the utility threshold in the consume-produce function cp has been set to match the (agent-independent) utility of equilibrium stocks under "special" prices.

3.4.3 The function h

The function h computes, for each agent, its average trading fitness

$$h\ [(f_1,x_1),\ldots,(f_{n_r},x_{n_r})]\ a = \frac{1}{n_r} * \sum_{k=1}^{k=n_r} f_k\ a\ . \quad (30)$$

Remember that, at the t-th prices iteration, the fitness achieved by an agent in a trading round is a random event: the set of interactions which take place in a round is defined by the random trading schedule of that round. At each round, the trading schedule is drawn from the set of all feasible trading schedules with equal probability. Moreover, each round is played starting from the same initial allocation x_0, zero fitness and constant agent-specific prices $y\ t$.

Therefore, $h\ a$ can be interpreted as an approximation of the expected trading round fitness obtained with the prices $y\ t\ a$.

[22] In [8], we study a simple class of models and suggest an explanation for this behavior.

4 Model implementation, numerical experiments

We have implemented the prices iteration (1) on the basis of the specification presented in the previous sections.

The implementation has been done in C++. It relies on standard libraries and on the collection of generic software components SC [2]. C++ is a statically typed programming language. Thus, when developing programs from specifications, one has to choose which specifications to enforce at compile time and which at run time.

In practice, the capabilities of expressing specifications at compile time are limited by the lack of explicit language support for higher order functions, constrained genericity and non-integer value genericity. Concept-based language extensions [6] are a promising approach but need to be practiced and substantiated with a better understanding of dependent types.

We have taken a pragmatic approach and enforced most specifications at run time. This has been done mainly through pre- and post-conditions clauses following a design-by-contract (DBC) [9] approach.

Of course, such approach can not guarantee that the implementation is correct that is, that it fulfills the specification. We turn back to this problem in section 4.2. In the rest of this section we discuss a number of results obtained with this implementation. All results have been obtained for 6 goods $G = \{g_1,\ldots,g_6\}$ and 600 agents $A = \{a_1,\ldots,a_{600}\}$. Thus, the number of sectors is equal to 6 and the number of agents per sector n_a is 100. The other parameters are defined according to section 3.1 and to the following setup:

- the sector function s is

$$s\ a_i \;=\; g_j,\ j = 1 + (i-1)/100, \quad i = 1,\ldots,600\ .$$

- The sector-to-sector number of peers, np, is 5.
- The initial allocation x_0 is defined by equation (28) with

$$x'_0\ g_j = \frac{|G|}{|A|} * (1+j), \quad j = 1,\ldots,6\ .$$

The correspondent special prices (29) normalized by g_1 are

$$\frac{\hat{p}\ g_j}{\hat{p}\ g_1} = \frac{2}{j+1}\ . \quad (31)$$

- The copy-mutate fraction cmf is 0.05.
- The mutation probability mp is 0.1.
- The mutation factor mf is 0.95.
- The number of trading rounds per prices iteration n_r is 10.

For any agent $a \in A$, the initial price $y_0\ a\ g_1$ is set to one. All other prices are drawn randomly from $(0,1]$ with uniform probability distribution.

The above setup is, to the best of our knowledge, the one used to compute the results shown in figures 1 and 2 of [5][23]. These results suggest that, under (1)

- the standard deviations of $y\ t$ rapidly decline in the first 1500 trading rounds that is, at 10 rounds per iteration, for t between 0 and 150.

[23] In [5] the mutation probability is reported to be 0.01 that is, one tenth of mp. However, the value used in the original Delphi implementation is equal to mp.

- at large times — for numbers of trading rounds (iterations) of the order of 10^5 (10^4) — the agent-specific prices tend to converge[24] towards the special prices (31).

In the next sections we compare these results with those obtained with our implementation. Two caveats are at place:

First, as it turns out, the dynamics of prices at large times depends critically on the conditions upon which the trading fitness f is incremented. In Gintis' original program, this is done only upon "successful" trades. The source code provided by Gintis shows that trades are taken to be be "successful" even when the amounts of goods actually exchanged by the two interacting agents are zero. This seems to be inconsistent. In our implementation, the trading fitness f is incremented if (and only if) an elementary bilateral trade between two agents yields non zero exchanges. More precisely, *cs* is executed only if *trp* returns non zero exchanges or, equivalently, if $x' \neq x$ in (11):

$$\delta_1 > 0 \ \vee \ \delta_2 > 0 \tag{32}$$

Second, in the original implementation exchanges are always rationed according to specifications (22) and (23). Rationing or, in other words, demand-limited trade resolving policies are certainly necessary[25] for sequences of elementary bilateral exchanges at fixed equilibrium prices to yield equilibrium allocations, see [3]. On the other hand, the analysis presented in [8] suggests that, at large times, rationing might prevent the convergence of agent-specific prices towards stochastically stable equilibrium prices. As anticipated, our implementation allows for rationing (and, in fact, production and consumption) to be disabled.

4.1 Short times price dynamics

In [5], the dynamics of prices at short times is described in terms of three standard deviations: a "cross-period standard deviation of mean private prices" and "inter-agent standard deviations of private producer and consumer prices".

Unfortunately, [5] does not define operational rules for computing these quantities and the vertical scale in figure 1 of [5] suggests that these quantities are not standard deviations or that the probability distribution for the initial prices is not uniform. Here, we describe the dynamics of prices at short times in terms of two mean squared deviation functions. The first one simply computes, for any price, the mean squared deviation over all agents for that (normalized) price:

$$msd : (A \to P) \to G \to \mathbb{R}$$

$$msd\ y\ g = \frac{\sum_{a \in A} \left(\frac{y\ a\ g}{y\ a\ g_1} - \mu\ y\ g \right)^2}{|A|}, \quad \mu\ y\ g = \frac{\sum_{a \in A} \frac{y\ a\ g}{y\ a\ g_1}}{|A|}$$

The second function computes, for any sector, the sector mean of the mean squared deviation of the prices of that sector:

$$msd' : (A \to P) \to G \to \mathbb{R}$$

$$msd'\ y\ g = \frac{\sum_{g' \in G} * \left(\frac{1}{|s^{-1}\ g|} * \sum_{a \in s^{-1}\ g} \left(\frac{y\ a\ g'}{y\ a\ g_1} - \mu'\ y\ g\ g' \right)^2 \right)}{|G| - 1}, \tag{33}$$

$$\mu'\ y\ g\ g' = \frac{1}{|s^{-1}\ g|} * \sum_{a \in s^{-1}\ g} \frac{y\ a\ g'}{y\ a\ g_1}\ .$$

Notice that, because of the normalization, the mean and the mean squared deviation of g_1 are 1 and 0 on any subset of $|A|$. This is why, in (33), the sum over $g' \in G$ is divided by $|G| - 1$ and not by $|G|$.

In figure 1 we report the graphs of $msd\ y$ (top) and $msd'\ y$ (bottom). Thus, the curves on the top are mean squared deviations (over all agents) of prices, one curve for each price. Since the prices are normalized, the mean squared deviation for g_1 is identically zero.

In contrast, the curves on the bottom are averages (over prices) of the mean squared deviations of the prices of a given sector, one curve for each sector. As one would expect, the sector-specific average mean squared deviations decrease faster than the mean squared deviations taken over all agents, especially at very short times. This is because the copy-mutate rule *cm* is applied sector-wise, see section 3.3.

Notice that the rates at which mean squared deviations decrease for increasing number of trading games are different for different prices (top) and sectors (bottom). We do not have an explanation for this behavior. Also, notice that the results of figure 1 seem qualitatively different from those reported in figure one of [5]. We come back to this point in the next section.

4.2 Large times price dynamics

In [5], the dynamics of prices at large times is described in terms of the relative deviations of average prices from the special prices (31). Such deviations are:

$$dev : (A \to P) \to G \to \mathbb{R}, \quad dev\ y\ g = -\frac{\mu\ y\ g - \frac{\hat{p}\ g}{\hat{p}\ g_1}}{\mu\ y\ g}$$

Again, $dev\ y\ g_1$ is identically zero. Figure 2 shows the mean squared deviations *msd* (top) and the deviations of the mean prices from the special prices *dev* (bottom) at large times. The results shown in figure 2 do not confirm those presented in [5]. While [5] does not report mean squared deviations at large times, the graphs of the deviations of the mean prices from the special prices on the right of figure 2 do not suggest that agent-specific prices tend to "converge" towards the special prices at large times.

Let us pause for a moment. We have derived a specification for the prices iteration (1) on the basis of [5] and of an original model implementation made available by the author. We have implemented (1) from such specification. Our numerical results fail to reproduce those presented in [5]. There are three logically possible reasons for such failure:

1. Our specification is a good description of the model presented in [5] but our implementation does not fulfill the specification, that is, it is not correct.

[24] of course, modulo stochastic mutations as effected through the copy-mutate rule *cm*.

[25] But, in general, not sufficient.

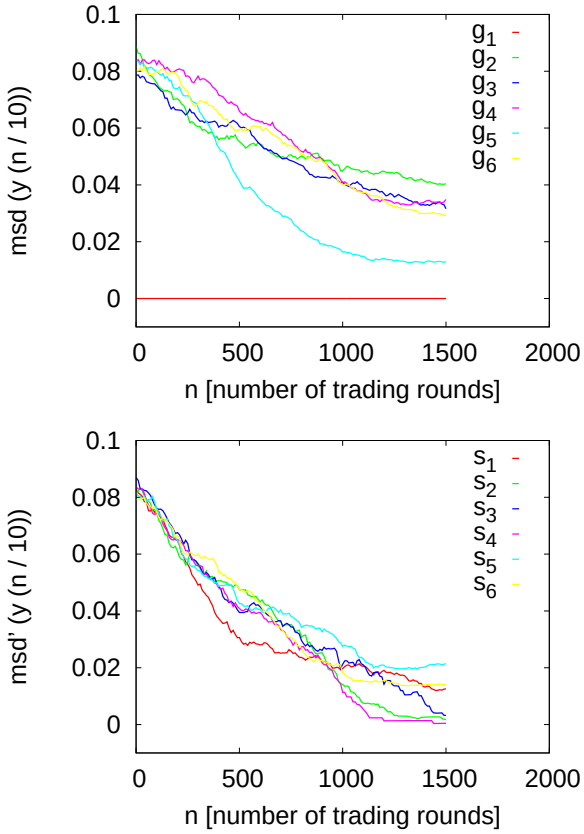

Figure 1. Standard setup: *msd* (top) and *msd′* (bottom).

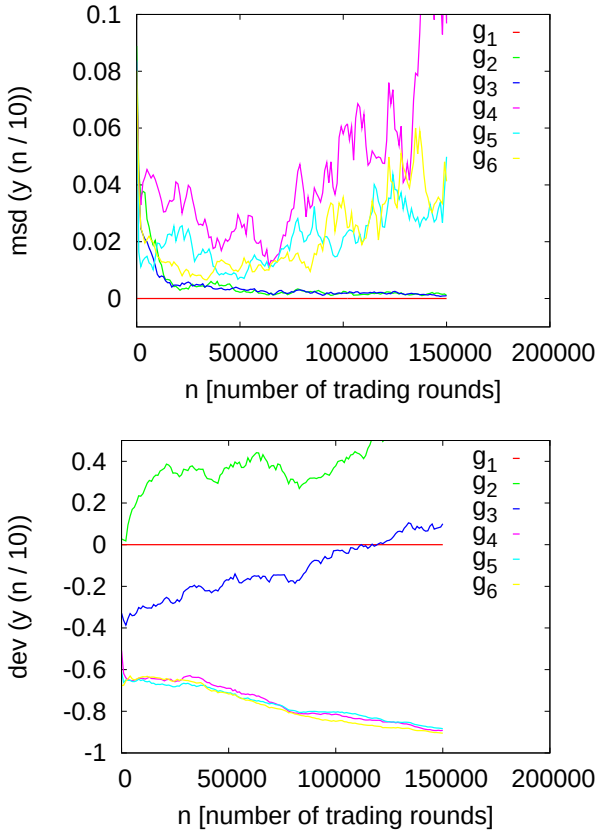

Figure 2. Standard setup: *msd* (top) and *dev* (bottom).

2. Or, our implementation is correct but our specification is not a good description of the model presented in [5].
3. Or, our model specification is a good description of the model presented in [5]. The results presented in [5], however, have been obtained with an implementation which does not fulfill such specification.

While 1 and 2 and 2 and 3 are mutually exclusive, it can of course be that both 1 and 3 are true: implementational errors never can be ruled out.

We can explain the inconsistencies between our numerical results and those presented in [5] by looking at the mechanisms that control the dynamics of prices at large times. We start by noticing that the results presented so far are robust with respect to perturbations of the copy-mutate rule *cm*[26] and focus the attention on the functions that define a trading game: *tr* and *cp*.

As anticipated above, the analysis presented in [8] suggests that, at large times, rationing might prevent the convergence of agent-specific prices towards stochastically stable equilibrium prices. Thus, an obvious experiment is to run a simulation in which rationing is disabled that is, (21)-(24) are replaced by

$$trp : P \to P \to \mathbb{R}_{\geq 0} \times G \to \mathbb{R}_{\geq 0} \times G \to \mathbb{R}_{\geq 0} \times G \to \mathbb{R}_{\geq 0} \times G \to \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$$

$$g_1 \neq g_2 \ \wedge \ trp\ p_1\ p_2\ (o_1, g_1)\ (d_1, g_2)\ (o_2, g_2)\ (d_2, g_1) = (\delta_1, \delta_2)$$

$$\Rightarrow$$

$$(\delta_1, \delta_2) = \begin{cases} (0,0) & \text{if } o_1 * (p_2\ g_1) < d_1 * (p_2\ g_2) \\ (\delta_1', \delta_2') & \text{otherwise} \end{cases} \tag{34}$$

$$(\delta_1', \delta_2') = \begin{cases} (x_2\ g_2, x_2\ g_2 * \frac{p_1\ g_2}{p_1\ g_1}) & \text{if } x_2\ g_2 < d_1 \\ (d_1, o_1) & \text{otherwise} \end{cases} \tag{35}$$

Except for this modification, the setup is the same as the of figure 2. Figure 3 shows the usual mean squared deviation (top) and relative deviations (bottom) graphs. As in [5] agent-specific prices appear to converge, at large times, towards the special prices. In fact, figure 3 bottom and figure 2 of [5] are quite similar. Before attempting at drawing any conclusions, let us consider the impact of two more controls on the dynamics of prices at large times.

First, remember that the consume-produce function *cp* is responsible for updating the trading fitness of interacting agents and for modeling consumption and production. As dis-

[26] of course, the speed at which the mean squared deviations of the prices decrease at short times and the amplitude of the high frequency oscillations in the graphs of *dev* depend on the values of cmf, mp and mf. In a number of numerical experiments (not reported here) designed to study how the dynamics of prices at large times depends on the values of cmf, mp and mf, however, we never observed small perturbations of these parameters (or of the function *cm*) to yield dramatic changes in the dynamics of prices. These results are consistent with those reported in [5].

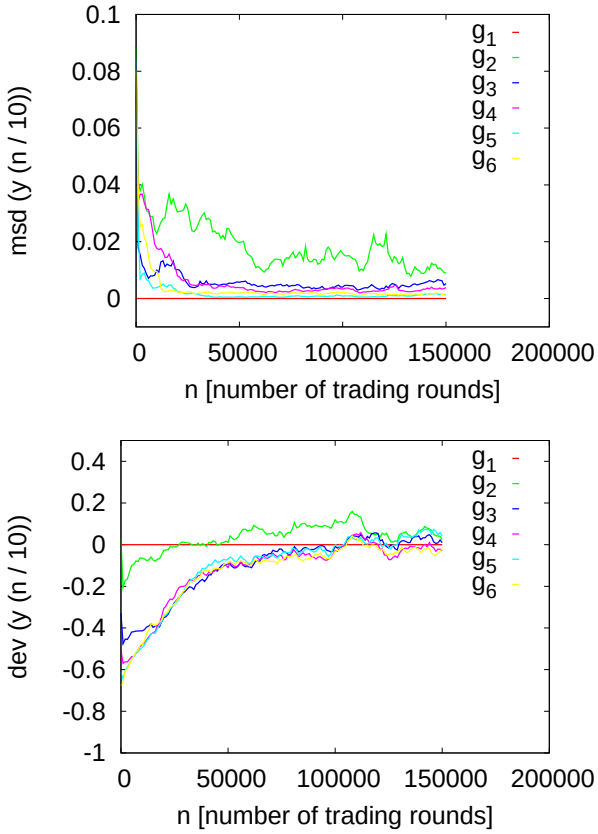

Figure 3. Standard setup but without rationing: *msd* (top) and *dev* (bottom).

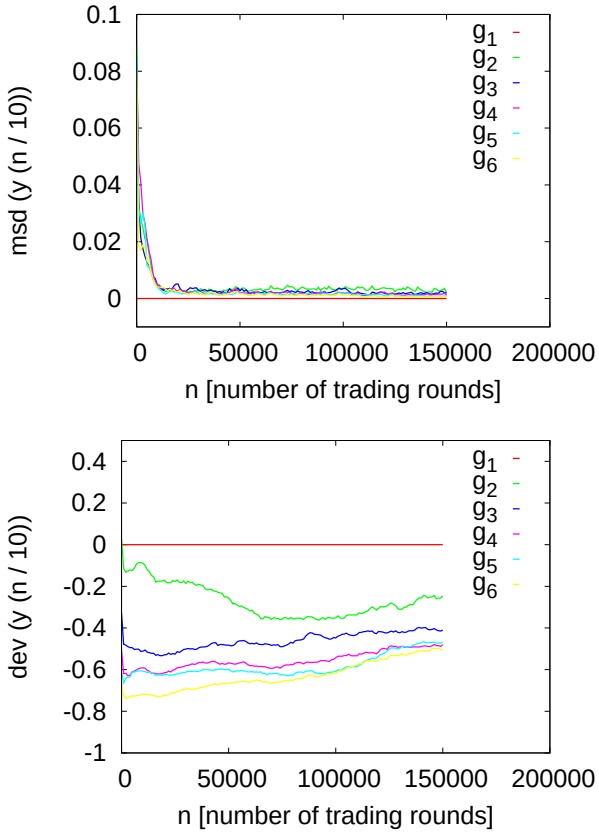

Figure 4. Standard setup but without consumption/production: *msd* (top) and *dev* (bottom).

cussed in section 3.4.2, consumption and production depend on a utility threshold which we find difficult to motivate. In a first experiment, we switch off consumption and production that is, we replace (26) with

$$a \in \{a_1, a_2\} \Rightarrow x'\ a = x\ a$$

Figure 4 reports the results obtained for the same setup of figure 2 but without consumption and production. The effect of switching off consumption and production is clearly visible in the graphs of the mean squared deviation of prices. As the number of iterations increases, the mean squared deviation of prices decrease remarkably. The prices, of course, tend to stabilize (again, modulo stochastic mutations) but not around the special prices (31): the deviations of the mean prices from the special prices (figure 2, bottom) do not tend towards zero as the number of iterations increases. On the other hand, disabling both consumption and production and rationing brings back the large times behavior of figure two with, not surprisingly, significantly lower mean squared deviations, see figure 5. Second, the conditions upon which the fitness of the interacting agents is incremented — the notion of "successful" trade discussed above — has of course an impact on the result of a trading game, the agents trading fitness f. A natural question is therefore whether such conditions critically affect the dynamics of prices at large times. Figures 6, 7 shows the results obtained with the same setups of figures 2, 3 but with modified conditions for incrementing the trading fitness. Specifically, in figures 6, 7 the trading fitness of the two agents entering an elementary bilateral trade is incremented if

$$o_2 > 0 \wedge d_2 > 0 \wedge x_2\ g_2 > 0 \wedge o_1 * (p_2\ g_1) \geq d_1 * (p_2\ g_2) \quad (36)$$

This condition can be easily derived from the original implementation. It is necessary for a non-trivial exchange to take place but, of course, not sufficient. Thus, the modified condition weakens our specification. A comparison between figure 2 and 6 suggests that, when bilateral trades are rationed, the conditions upon which the fitness of the interacting agents is incremented affect the dynamics of prices at large times significantly. In contrast, when bilateral trades are not rationed, the same conditions appear to impact the dynamics of prices less severely, see figures 3 and 7.

We have discussed the outcome of numerical experiments on the dynamics of prices at large times in terms of the graphs of *msd* and *dev* during the first 15000 iterations or 150000 trading games. Each computation, however, has been carried out for 1500000 iterations (15000000 trading games). The results (not shown here) confirm the observations done for the first 15000 iterations.

5 Conclusions, future work

We have applied the functional framework proposed in [3] to derive a "bona fide" mathematical specification of the model discussed in [5].

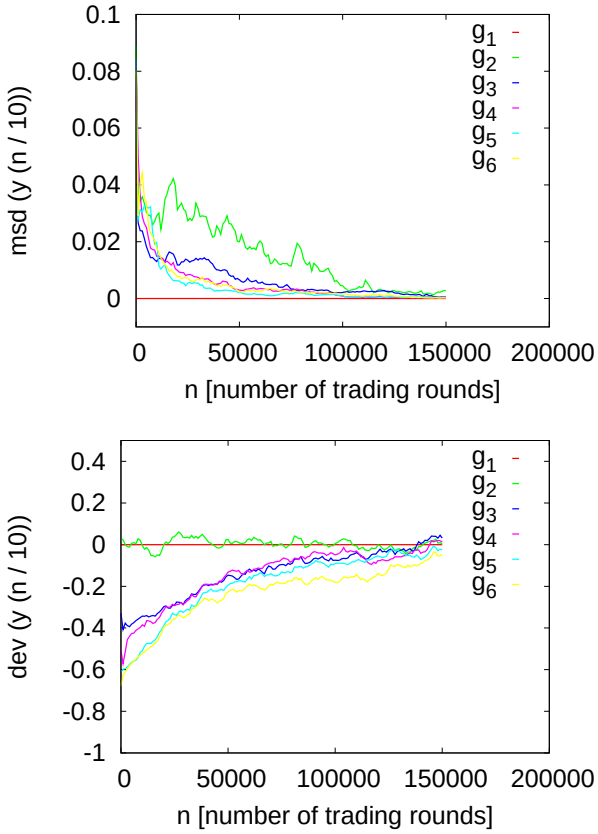

Figure 5. Standard setup but without rationing and consumption/production: *msd* (top) and *dev* (bottom).

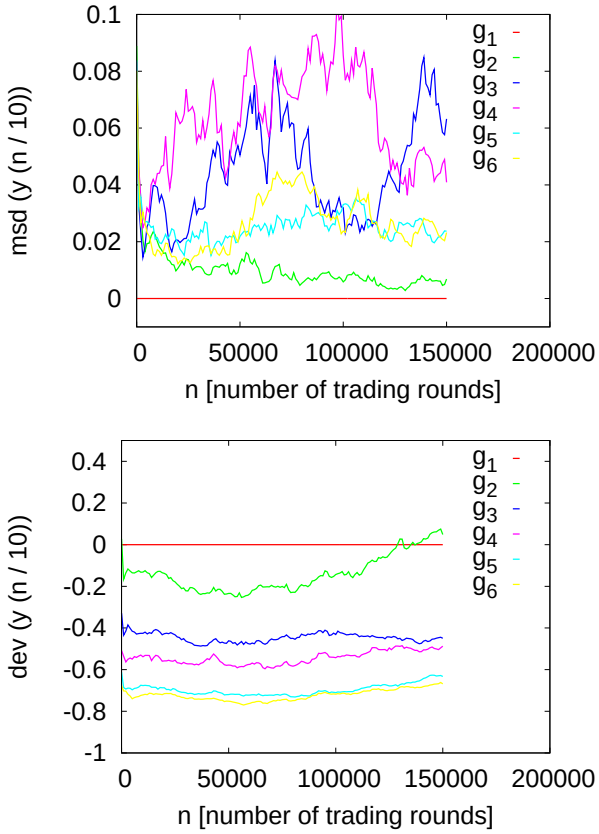

Figure 6. Standard setup but the trading fitness of the interacting agents is incremented if condition (36) (in contrast to (32)) is fulfilled: *msd* (top) and *msd′* (bottom).

The specification is based on the model description presented in [5], on the original model implementation in Delphi kindly made available by the author and on the Java / MASON model re-implementation presented in [4].

The specification is complete in the sense that it allows unambiguous model implementations. We have written one such implementations in C++.

We have used this implementation to run a number of numerical experiments on the dynamics of prices. These experiments partially confirms the results presented in [5]. In particular, we were able to independently reproduce the price dynamics reported in figure 2 of [5], see section 4, figure 3.

On the other hand, the numerical results reported in figure 2 and figure 3 suggest that the emergence of the special, quasi-public equilibrium prices (31) from agent-specific prices (the "private" prices of [5]) under evolutionary (imitation-mutation) dynamics is by no means a robust feature of the model presented in [5]. Price convergence critically depends on properties of the underlying trading game, in particular, of the trade resolving policy.

When bilateral exchanges between agents are rationed, agent-specific prices do not seem to converge towards the quasi-public equilibrium prices (31). This behavior is consistent with the analysis proposed in [8].

A comparison of figures 3, 4 and 5 also suggests that production and consumption induce higher price volatility but do not essentially affect the convergence of prices.

Moreover, figures 2, 6 and 3, 7 suggest that the notion of successful trade or, in other words, the conditions under which the trading fitness of the interacting agents is incremented can have a significant impact on the dynamics of prices. This is particularly true for the case in which bilateral trades are rationed.

Of course, the above "conclusions" can only be preliminary. They need to be confirmed by independent numerical experiments and substantiated by analytical results. On the other hand, the model specification derived in section 3 and, in particular, the specification of the trading policies *odp* and *trp* allow one to draw logical consequences which are independent of numerical results.

One consequence is that rationing constraints are necessary (but certainly not sufficient) for sequences of elementary bilateral trades *tr* at fixed, constant equilibrium prices to drive $x0$ towards the equilibrium allocation correspondent to the given prices. We have discussed this issue in detail in section 4.2 of [3] but the reason why rationing is necessary is obvious: if the amounts of good exchanged in a trade exceed, e.g., the demand of the responder, this is going to be left with an amount of good in excess of its optimal value. Since elementary bilater trades only allow agents to decrease the amount of their sector specific "offer" good, there is no way for the responder to achieve its optimal stock.

In other words, under elementary bilateral trades, optimal

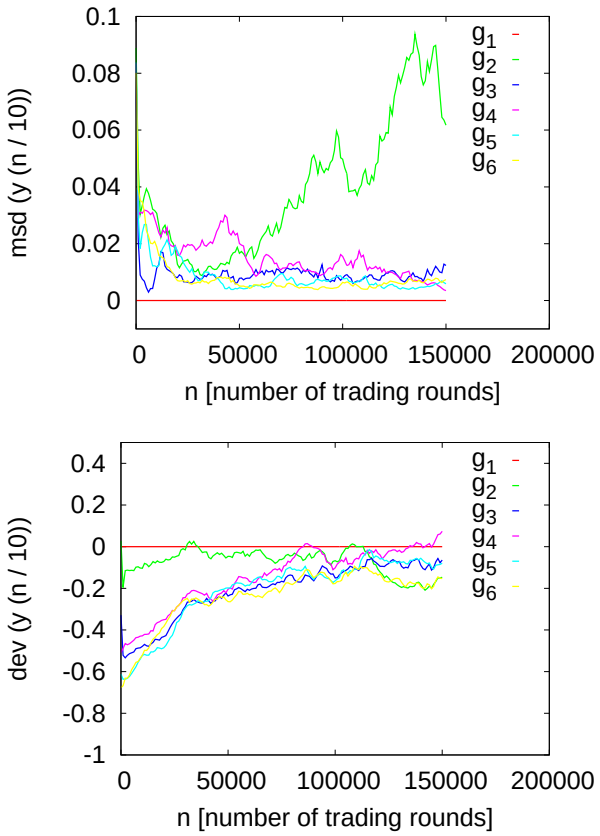

Figure 7. Standard setup but 1) without rationing and 2) the trading fitness of the interacting agents is incremented if condition (36) (in contrast to (32)) is fulfilled: msd (top) and msd' (bottom).

allocations can only be reached "from below" in the demand goods and "from above" in the offer good. Over-shootings (under-shootings) in the demand (offer) goods cannot be balanced by further interactions.

On the other hand, if rationing turns out to prevent convergence of agent-specific prices (as the numerical results shown in figure 2, 4 and 6 seem to suggest), we are forced to conclude that, in the model of exchange proposed in [5], convergence of allocations[27] and convergence of prices[28] are mutually exclusive.

This would mean that, even under the assumptions at the basis of the model presented in [5][29] we still do not know simple and plausible rules of bilateral interaction which guarantee emergence of equilibrium prices under imitation-mutation dynamics and, for fixed, exogenous equilibrium prices, convergence of allocations towards the correspondent equilibrium allocations.

A final word of precaution is needed. The numerical results presented in [5], [4] and in this paper have all been obtained with a constant utility profile and with the utility function (2), (3). This utility function supports infinitely many equilibrium prices-allocation pairs, see section 3.1 of [3].

To substantiate numerical conjectures on the dynamics of prices, numerical experiments of the kind reported in section 4 and in [5] should be run for less accommodating utility functions.

In particular, the dynamics of agent-specific prices at large times should be investigated for utility functions which support unique equilibrium prices and for the case in which the utility function is different in different sectors. Such investigations go beyond the scope of this paper.

[27] Towards equilibrium allocations, through sequences of elementary bilateral trades at fixed, given equilibrium prices.

[28] Towards quasi-public equilibrium prices, under imitation-mutation dynamics driven by the trading fitness achieved in sequences of elementary bilateral trades at fixed agent-specific prices.

[29] That is, offer and demand policies based on utility maximization, perfect knowledge of the stocks that maximize an agent's utility for every agent for arbitrary prices, fully deterministic elementary bilateral trades between agents.

ACKNOWLEDGEMENTS

The authors thank the reviewers, whose comments have lead to significant improvements of the original manuscript. The work presented in this paper heavily relies on free software, among others on hugs, vi, the GCC compiler, Emacs, LaTeX and on the FreeBSD and Debian / GNU Linux operating systems. It is our pleasure to thank all developers of these excellent products.

REFERENCES

[1] 'Ariane 5 Flight 501 Failure', Technical report, Report by the Inquiry Board, (1996). esamultimedia.esa.int/docs/esa-x-1819eng.pdf.

[2] N. Botta and C. Ionescu, 'Relation based computations in a monadic BSP model', *Parallel Computing*, **33**, 795–821, (2007).

[3] N. Botta, A. Mandel, C. Ionescu, M. Hofmann, D. Lincke, S. Schupp, and C. Jaeger, 'A functional framework for agent-based models of exchange', *Applied Mathematics and Computation*, **218**(8), 4025–4040, (December 2011).

[4] P. Evensen and M Märdin. An Extensible and Scalable Agent-Based Simulation of Barter Economics. Master Thesis. Chalmers Techn. Univ. & U. Gothenburg., 2009.

[5] H. Gintis, 'The emergence of a Price System from Decentralized Bilateral Exchange', *B. E. Journal of Theoretical Economics*, **6**, 1302–1322, (2006).

[6] Douglas Gregor, Jaakko Järvi, Mayuresh Kulkarni, Andrew Lumsdaine, David Musser, and Sibylle Schupp, 'Generic programming and high-performance libraries', *International Journal of Parallel Programming*, **33**(2), (June 2005).

[7] *Agent Based Models for Economic Policy Advice*, eds., B. LeBaron and Winker P., Lucius, 2008.

[8] A. Mandel and N. Botta, 'A Note on Herbert Gintis' "Emergence of a Price System from Decentralized Bilateral Exchange"', *The B.E. Journal of Theoretical Economics*, **9**, (2009).

[9] B. Meyer, *Object-oriented Software Construction*, Prentice-Hall Resource, Prentice-Hall, second edn., 2000.

[10] B. H. Mitra-Kahn. Debunking the Myths of Computable General Equilibrium Models. Schwartz Center for Economic Policy Analysis working Paper 2008-1, 2008.

[11] P. Taylor and L. Jonker, 'Evolutionarily Stable Strategies and Game Dynamics', *Mathematical Biosciences*, **40**, 145–156, (1978).

[12] *Handbook of Computational Economics II: Agent-Based Computational Economics*, eds., L. Tesfastion and K. Judd, North-Holland, 2006.

[13] Sherry Tuckle, *Simulation and Its Discontents*, MIT Press, 2009.