Applied Mathematics and Computation 218 (2011) 4025–4040

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A functional framework for agent-based models of exchange

N. Botta [a,*], A. Mandel [b], C. Ionescu [a], M. Hofmann [c,a], D. Lincke [a], S. Schupp [d], C. Jaeger [a]

[a] Potsdam Institute for Climate Impact Research, Germany
[b] Centre d' Economie de la Sorbonne, CNRS et Université Paris 1 Panthéon Sorbonne, France
[c] Institut für Mathematik, Freie Universität Berlin, Germany
[d] Institute for Software Systems, Hamburg University of Technology, Germany

ARTICLE INFO

Keywords:
Barter economies
General equilibrium
Agent-based methods
Functional specifications
Price dynamics

ABSTRACT

We propose a functional framework for studying agent-based dynamical models of exchange. The framework consists of a notation and of a number of elementary definitions. We argue that – in comparison to narrative descriptions – the framework has a number of advantages: (1) it allows one to express precisely the relationships between the classical notion of general equilibrium and agent-based models of exchange; (2) it supports computational descriptions of models of exchange which can be unambiguously implemented; (3) it assists the numerical investigation of such models by providing a specific set of elementary notions and computational primitives. These can be used to deduce model properties and to setup "crucial" numerical experiments for validating model implementations. In a companion paper, we apply the framework to specify and study a model of exchange in which multiple equilibrium prices coexist and prices evolve according to a simple trading scheme and to a generic genetic rule.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

1.1. Preliminaries

The idea of exchange has been at the core of economic modeling at least since Walras [43]. In Walrasian economies, a number of economic actors own certain quantities of goods. Goods have different types and can be freely exchanged between actors. Good prices emerge as ratios at which goods of a given type are exchanged for goods of another type.

The most influential mathematical formulation of this idea was probably the notion of general equilibrium proposed by Debreu [12]. While influential, the notion of general equilibrium as a mathematical model of models of exchange has been confronted with a number of criticisms.

These range from a critique of the "individual rationality" assumption implicit in the notion of general equilibrium [22,24,26,18] to the skepticism, raised by many authors after the Sonnenschein–Mantel–Debreu "anything goes" result [39], about the usefulness of the notion of general equilibrium for decision making and, more recently, for policy advice [25,15,13,31,32,10].

In the last two decades, a number of alternative formulations of the idea of exchange have been proposed as *computer-based* models [3]. More recently, so-called *agent-based* models of exchange have been proposed, see [41]. While a precise, established notion of agent-based models is lacking, there seems to be some shared understanding that agent-based models are particular computer-based models. Gintis, for instance, maintains that agent-based models are computer simulations of

* Corresponding author.
E-mail address: botta@pik-potsdam.de (N. Botta).

0096-3003/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2011.08.051

certain games, see [19, p. 2]. For the rest of this paper and where there is no risk of misunderstanding, we will often use "model of exchange" as an abbreviation for "agent-based model of exchange".

While computer-based models and, in particular, computer-based models of exchange have been applied to the simulation of "real" economies with some success, see [41], their usage in critical applications such as decision making and policy advise is still uncommon.

1.2. Computer-based models, exploratory programming

When considering computer-based models it is important to distinguish between models which are developed to solve well defined problems and models which are obtained through *exploratory programming* [38].

Examples of the first class of computer-based models are often found in scientific computing and engineering. Here, the problems to be solved are often well understood and can be stated precisely. Computer-based algorithms deliver (usually approximate) solutions to such problems. The accuracy of such solutions can be measured without resorting to empirical data. The availability of well defined problems allows model developers and implementers to design *crucial* experiments. These are experiments for which a negative outcome unambiguously indicates errors in the algorithm or in its implementation.

As an example, let $r2 \in Real \times Real \times Real \to Complex$ be a program which implements an algorithm for finding roots of quadratic equations of the form $ax^2 + bx + c = 0$. A suitable requirement for $r2$ could be

$$\forall\ a, b, c \in Real\ a \neq 0\ \wedge\ y = r2\ (a, b, c)\ \Rightarrow\ |ay^2 + by + c| \leqslant \epsilon, \tag{1}$$

where $\epsilon \in Real$ is a suitable residuum upper bound. Conversely, a tuple $a^*, b^*, c^* \in Real, a^* \neq 0$ for which $|a^*y^2 + b^*y + c^*| > \epsilon$ where $y = r2\ (a^*, b^*, c^*)$ would unambiguously indicate that $r2$ does not fulfill the requirement (1). This leaves two possibilities open:

- $r2$ is a wrong implementation of a correct algorithm for finding roots of quadratic equations of the form $ax^2 + bx + c = 0$.
- The algorithm for finding roots of quadratic equations of the form $ax^2 + bx + c = 0$ is not correct.

Of course, we can rule out the second possibility by choosing an algorithm for finding roots of quadratic equations which has been proved to be correct. In programming, requirements like (1) are often called *specifications*. In the example above (1) is a specification for computer-based models for finding roots of quadratic equations of the form $ax^2 + bx + c = 0$. The example makes clear that

(1) When considering computer-based models which are designed to solve well defined problems, it is useful to distinguish between three notions: the specification, the algorithm and its implementation.
(2) Specifications can be expressed in a clear mathematical notation. They are often written in terms of equations. Often such equations are given special names like problem equations, governing equations, governing laws.
(3) Though expressed in mathematical terms, specifications necessarily rely on notions and values which can depend on the computing architecture: in (1), for instance *Real* represents a (computing architecture dependent) set of floating point numbers. The upper bound ϵ is a value which also meaningfully depends on the sets *Real* and *Complex*. In many cases it is convenient to trade accuracy of specifications for intuitiveness and replace (1) with

$$\forall\ a, b, c \in \mathbb{R}\ a \neq 0\ \wedge\ y = r2\ (a, b, c)\ \Rightarrow\ ay^2 + by + c = 0. \tag{2}$$

This is the approach taken throughout this paper. Notice, however, that (2) is just a suggestive, convenient replacement for the original specification (1). Crucial experiments and proofs of incorrectness have to be based on this specification.

(4) Algorithms are often expressed in pseudo programming languages [2]. These are more difficult to read and analyze than specifications but still more readable than implementations. Implementations are written in programming languages. *Imperative* programming languages, e.g. C, C++, FORTRAN, Java, trade understandability for efficiency: the lack of referential transparency[1] makes reasoning on implementations difficult and error-prone. Functional programming languages, e.g. Haskell [16], close the gap between specifications and implementations and support model analysis, e.g., via equational reasoning. For most realistic applications, however, the run-time efficiency of functional languages is unacceptable.
(5) Ideally, developers of computer-based models would like to derive, possibly automatically, implementations from specifications. Alternatively, they would like, at least for critical applications such as air traffic control, financial markets or policy advise, to prove that actual implementations are correct, i.e., they fulfill their specification. These goals are often too ambitious for complex computer-based models. A realistic alternative is to implement computer-based models by combining software components which are themselves correct or, at least, by combining software components for which crucial experiments in the sense made clear above can be set-up straightforwardly.

[1] Referential transparency is a property of implementations. Implementations are said to be referentially transparent whenever expressions can be replaced with their values. Referential transparency is therefore a precondition for applying substitution rules in the study of programs. Implementations which are not referentially transparent are often said to be referentially opaque.

The development of computer-based models through exploratory programming,[2] is quite common in academic research and, in particular, in agent-based economic modeling. Here the problems are, especially in the beginning, often not well understood: precise specifications are not available and it is not clear which software components would be needed for (non-exploratory) implementations.

Notice that prototype implementations can themselves be seen as unambiguous, precise model descriptions which, however, expose too many details of the underlying computational model,[3] lack conciseness, expressivity, readability and are almost inaccessible to non-programmers.

Thus, prototype implementations are not very useful for explaining, communicating and studying computer-based models. They are usually complemented by other model descriptions. These often consist of a blend of narrative, mathematical equations and, perhaps, pseudo code. In principle, such descriptions could be precise enough to allow independent model re-implementations [21]. In practice, however, model re-implementation on the basis of narrative descriptions is nearly impossible. For consistent, independent model re-implementation, one needs unambiguous mathematical specifications.

Deriving mathematical specifications from prototype implementations and narrative descriptions is then the next step in model development. Once available, mathematical specifications build the basis for re-factoring early model prototypes and for model analysis and dissemination.

1.3. Goal, motivation

In this paper we propose a functional framework for specifying dynamical models of exchange. Concretely, we address 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.

Our work has been very much inspired by the studies of Herbert Gintis on price formation and price selection mechanisms, in particular by the model presented in [19]. The model is remarkable in two ways. On one hand, the results reported in [19] have raised a number of intriguing questions about price formation/selection mechanisms and about the relation between prices in agent-based dynamical models of exchange built around the notion of bilateral trades and general equilibrium prices [12].

On the other hand, the model seems to be simple enough to be easily understandable and re-implementable. Moreover, the author had provided, together with the model description presented in [19], a model implementation written in Delphi Pascal. Alas, our first attempts at reproducing the results presented in [19] with an independent model implementation written in C++ failed.[4]

As it turned out, the failure was not due to implementational errors but to a poor interpretation of the original model description. With the functional framework introduced in Section 3 we can now specify models of exchange of the kind described in [19] in a concise, unambiguous form and derive consistent model re-implementations.

1.4. Outline of the paper

The paper is organized as follows: in Section 2 we introduce the notation that will be used throughout the paper. The functional framework for dynamical models of exchange is presented in Section 3. We first rehearse the notions of general equilibrium, demand, excess demand in the context of exchange economies. These notions are at the core of "demand-driven" models of exchange. In Sections 3.3, 3.4, 3.5 we introduce the notions of bilateral exchange, model of exchange, bilateral trade, sector and trading schedule and round. These are the notions used to introduce trading games and dynamical models of exchange in Section 3.6.

In Section 4 we demonstrate how the functional framework can be applied to formulate unambiguously a number of questions about the relations between dynamical models of exchange and general equilibrium prices and allocations. We briefly discuss how the framework can be used to derive simple crucial experiments for validating model implementations.

2. Notation

In the context of agent-based models of exchange, two notions are particularly important: that of a set of agents and that of a set of goods. We denote these sets by A and G, respectively. Throughout this paper, we assume A and G to be finite and we formulate economic notions mainly in terms of functions. Often, these functions take values in A or G.

Stocks, for instance, are formulated as functions of type $G \to \mathbb{R}_{\geqslant 0}$: they associate to the goods in G non-negative real numbers. Thus, we write

$$q \in G \to \mathbb{R}_{\geqslant 0}$$

[2] Exploratory programming is also called software prototyping.

[3] In implementations in imperative languages without garbage collection, for instance, memory management details.

[4] In [6], we give a detailed account of such failure.

to posit that q is a stock. Similarly, allocations are functions which associate stocks to the agents in A and we write

$$x \in A \to (G \to \mathbb{R}_{\geqslant 0})$$

to posit that x is an allocation. We denote function application by juxtaposition, following a common usage in category theory where "evaluation is a kind of composition" [27]: $x\ a \in G \to \mathbb{R}_{\geqslant 0}$ is the stock of a according to x and $x\ a\ g \in \mathbb{R}_{\geqslant 0}$ the quantity of g according to $x\ a$.

This notation is standard in mathematics and computing science but not quite common in engineering and economics. In economics, stocks, for instance, are often formulated in terms of vectors in $\mathbb{R}^n_{\geqslant 0}$ where $n = |G|$ and one writes $q \in \mathbb{R}^n_{\geqslant 0}$ and q_j instead of $q \in G \to \mathbb{R}_{\geqslant 0}$ and $q\ g$. While popular, this approach has a number of disadvantages.[5] We argue that a functional notation is clearer and better suited for writing specifications.

In much the same way as stocks and allocations, prices, utilities and utility profiles are formulated through functions of type $G \to \mathbb{R}_{>0}$, $(G \to \mathbb{R}_{\geqslant 0}) \to \mathbb{R}$ and $A \to ((G \to \mathbb{R}_{\geqslant 0}) \to \mathbb{R})$, respectively. To keep the notation terse, we introduce the following type synonyms for stocks, prices and utilities:

$$\begin{aligned} Q &= G \to \mathbb{R}_{\geqslant 0}, \\ P &= G \to \mathbb{R}_{>0}, \\ U &= Q \to \mathbb{R}. \end{aligned}$$

With these abbreviations, prices, utilities, allocations and utility profiles are functions in Q, P, U, $A \to Q$ and $A \to U$, respectively.

3. A functional framework for dynamical models of exchange

In this section we introduce a framework for specifying and studying dynamical models of exchange. As anticipated in the introduction, we aim at developing *mathematical* specifications. This is in contrast to prominent non-mathematical specification methods such as UML [17], user-stories [9] and check-list driven methods recently introduced to improve the understandability of agent-based models [21,35].

At the same time, the approach presented here is more coarse grained than those based on refinement calculus [14,33,4] and certainly less demanding than approaches based on formal specification languages like Z [7], VDM, B [23,1], Maude [8] or constructive type theory [28,34].

The framework can be understood without mathematical or computing science notions more advanced than those introduced in the previous section. However, the framework takes a computing science viewpoint on computational economics and some acquaintance with functional programming languages might be of advantage.

While we have carefully avoided constructs from specific languages, the notation is obviously influenced by Haskell. This is particularly evident in the usage of juxtaposition for function application and of currying.[6] Because of the Haskell-like notation and of the systematic usage of functions (in contrast to, e.g., vectors) for the representation of properties of finite sets – prices of goods, for instance – we call the proposed framework *functional*.

3.1. Walrasian equilibrium $\mathcal{E}$

An allocation $x \in A \to Q$ and a price $p \in P$ are in equilibrium with respect to an allocation $x_0 \in A \to Q$ and a utility profile $u \in A \to U$ iff [42]:

$$\forall\ g \in G, \quad \sum_{a \in A} x\ a\ g = \sum_{a \in A} x_0\ a\ g, \tag{3}$$

$$\forall\ a \in A, \quad x\ a \in \underset{y \cdot p \leqslant (x_0\ a) \cdot p}{\operatorname{argmax}} u\ a\ y. \tag{4}$$

We write $(x, p)\mathcal{E}(x_0, u)$. The $|G|$ equations of (3) formulate conservation of goods: for every good in G, the total amount of that good in x has to be equal to the total amount in x_0. In other words, x is a re-allocation of x_0 (and vice versa).

[5] For instance, when dealing with functions of more than one variable – multi-dimensional arrays – or when developing specifications. In the latter case, replacing finite functions with vector-based representations leads to an obvious loss of information: for $G \neq G'$ but $|G| = |G'|$, $G \to X$ and $G' \to X$ are different sets but $X^{|G|}$ and $X^{|G'|}$ are not. This is not a problem whenever the information which is missing in the representation is clear from the context. Specifications, however, are particularly useful exactly when this is not the case: they aim at making the context visible, not at hiding information.

[6] Currying (after Haskell Brooks Curry) refers to the representation of functions of more than one variable as functions of one variable which, in turn, return function values. For instance, a function f from $A \times B$ into C is represented, in curried form, by a function f' which takes values in A and returns functions from B to C:

$$\begin{aligned} &f \in A \times B \to C, \\ &f' \in A \to (B \to C), \\ &f'\ a\ b = f\ (a, b). \end{aligned}$$

The $\to$ is taken to be right-associative and $A \to (B \to C)$ is written as $A \to B \to C$.

The $|A|$ equations of (4) are optimality conditions. For any agent in A, they require x to maximize the utility of that agent under a budget constraint. In such constraint, $y \cdot p$ represents the scalar product between y and p

$$y \cdot p = \sum_{g \in G} (y\ g) * (p\ g). \tag{5}$$

We say that $y \cdot p$ is the value of the stock y according to the prices p. When considering the equilibrium relation, it is important to notice that the prices p enter Eq. (4) only in the budget constraint and as linear factors of both sides of the inequality. Thus,

$$(x,p)\mathcal{E}(x_0,u)\ \wedge\ \lambda \in \mathbb{R}_{>0}\ \Rightarrow\ (x,\lambda * p)\mathcal{E}(x_0,u).$$

This implies that equilibrium prices, if they exist, can only be unique up to a multiplicative factor. In other words: what matters in the notion of equilibrium are not the prices of the single goods but the ratios between such prices. Therefore, methods for computing (x,p) given (x_0,u) always deliver *normalized* prices. These are obtained by dividing all prices by the price of a reference good, say $g_r \in G$.[7] This is always possible because prices are strictly positive.

Another notion which is worth keeping in mind when considering re-allocations, is that of value-preserving transformations. Let x'_0 be a re-allocation of x_0:

$$\forall\ g \in G,\quad \sum_{a \in A} x'_0\ a\ g = \sum_{a \in A} x_0\ a\ g.$$

Moreover, let the value of $x'_0\ a$ according to p be equal to the value of $x_0\ a$ according to p for all $a \in A$. Then, obviously

$$(x,p)\mathcal{E}(x_0,u)\ \Rightarrow\ (x,p)\mathcal{E}(x'_0,u).$$

This suggests that value-preserving exchanges of goods between agents – exchanges are special re-allocations – might be suitable primitives for designing iterative methods for computing equilibrium allocations given x_0 and equilibrium prices p. We come back to this idea in Section 4.

The theory of general equilibrium uses fixed-point theorems to provide sufficient conditions for an equilibrium to exist, see [12]. While existence is granted under fairly general conditions, uniqueness requires very restricting conditions, see [29]. A prominent example of a utility profile which supports multiple equilibria is the one originally proposed by Scarf [37]:

$$u\ a\ y = \min_{g \in G} (y\ g)\ /\ (w\ g), \tag{6}$$

where w is a function of type $G \to \mathbb{R}_{>0}$ like the prices. When the total quantity of every good in x_0 is positive and w is a multiple of these total quantities

$$w = \lambda * \sum_{a \in A} x_0\ a\ \wedge\ \lambda \in \mathbb{R}_{>0}, \tag{7}$$

any $p \in P$ and $x \in A \to Q$ s.t.

$$x\ a = \frac{(x_0\ a) \cdot p}{w \cdot p} * w, \tag{8}$$

are in equilibrium with (x_0,u), see Appendix A.

3.2. Demand, excess demand

The notion of equilibrium can be used to formulate a variety of problems. A natural one is that of computing a pair (all pairs) (x,p) which is (are) in equilibrium with some given allocation x_0 and utility profile u. This is not, in general, the problem addressed by dynamical models of exchange.

In a simplified setup, however, methods for solving this problem rely on the notions of *demand* and *excess demand*. These, in turn, are often used for defining models of exchange. We therefore formulate the simplified setup and rehearse the notions of demand and excess demand.

Consider pairs (x_0,u) in $(A \to Q) \times (A \to U)$ such that the optimality equation (4) have unique solutions for arbitrary prices.[8] For such pairs, the function

$$\begin{aligned}
&d\ x_0\ u \in A \to (P \to Q)\\
&\forall\ a \in A,\quad \forall\ p \in P,\quad q = d\ x_0\ u\ a\ p\ \Rightarrow\\
&\qquad\qquad q \cdot p \leqslant (x_0\ a) \cdot p\\
&\qquad\qquad \wedge\\
&\qquad\qquad \forall\ q' \in Q,\quad q' \cdot p \leqslant (x_0\ a) \cdot p\ \wedge\ u\ a\ q' \geqslant u\ a\ q\ \Rightarrow\ q' = q
\end{aligned}$$

[7] In economics, such reference good is called *numéraire*.

[8] The Scarf utility profile (6) and allocations x_0 which fulfill (7) are examples of such pairs.

is called the *demand profile* and $d\ x_0\ u\ a\ p \in Q$ is called the demand of a under prices p. Obviously,

$$(x,p)\mathcal{E}(x_0,u)\ \Rightarrow\ x\ a = d\ x_0\ u\ a\ p,\quad \forall\ a \in A.$$

Similarly, $d\ x_0\ u\ a\ p - x_0\ a \in Q$ is called the *excess demand* of a under prices p.

When the demand profile for given (x_0,u) is known analytically, the problem of computing equilibrium stocks and prices boils down to the problem of solving $|G|$ (generally non-linear) equations for the unknown equilibrium prices $p\ g$ for $g \in G$:

$$\forall\ g \in G,\quad \sum_{a\in A}(d\ x_0\ u\ a\ p\ g - x_0\ a\ g) = 0.$$

This formulation of goods conservation (3) in terms of the agents demand functions shows that equilibrium prices, if they exist, are those prices for which the total excess demand is zero for every good. This explains why equilibrium prices are often said to "clear the market" or to "coordinate the agents demands".

3.3. Bilateral exchanges

As mentioned in the introduction, the notion of exchange has a prominent role in economic modeling. Many models, in particular of barter economies, rest on the notion of *bilateral exchange*. We formulate the notion of bilateral exchange through a relation between allocations. We say that two allocations $x, x' \in A \to Q$ are related through a bilateral exchange between two agents $a_1, a_2 \in A$ iff:

$$x'\ a \neq x\ a\ \Rightarrow\ a = a_1\ \vee\ a = a_2, \tag{9}$$

$$x'\ a_1 - x\ a_1 = x\ a_2 - x'\ a_2. \tag{10}$$

We write $x(\mathcal{X}\ a_1\ a_2)x'$. In the above specification, $x'\ a_1\ g - x\ a_1\ g$ is the amount of good g exchanged between a_1 and a_2. Obviously, allocations which are related by a bilateral exchange are re-allocations of each other:

$$x(\mathcal{X}\ a_1\ a_2)x'\ \Rightarrow\ \forall\ g \in G,\quad \sum_{a\in A} x'\ a\ g = \sum_{a\in A} x\ a\ g.$$

In decentralized models of exchange [19], pairs of agents engage in interactions which result in *elementary* bilateral exchanges. These are exchanges of exactly two goods: we say that two allocations $x, x' \in A \to Q$ are related through an elementary bilateral exchange of $g_1, g_2 \in G$ between two agents $a_1, a_2 \in A$ iff:

$$x(\mathcal{X}\ a_1\ a_2)x', \tag{11}$$

$$(x' - x)\ a_1\ g \neq 0\ \Rightarrow\ g = g_1\ \vee\ g = g_2. \tag{12}$$

We write $x(\mathcal{X}_e\ a_1\ a_2\ g_1\ g_2)x'$.

3.4. Models of exchange, elementary bilateral trades

As far as we are aware, there is not a well-established notion of model of exchange in economics. In this paper, we use the notion of (agent-based) model of exchange to denote models in which the outcome of interactions between agents are (possibly trivial) exchanges of goods. Other economic processes, e.g., production and consumption, are not present.

In particular, we are interested in models of exchange in which agents engage in elementary bilateral trades. These are interactions which result in elementary bilateral exchanges as specified in (11), (12).

Elementary bilateral trades are defined in terms of two *policies*: an agent-specific "offer and demand" policy and an agent-independent trade-resolving policy. We use the term policy in the sense of control theory: policies are functions that maps "states" into "actions" or controls.

In case of the offer and demand policy, the actions are two pairs in $\mathbb{R}_{\geqslant 0} \times G$. They represent the agent's offer and demand for two specific goods.[9] The state is agent-specific: in a dynamical model of equilibrium, for instance, the offer and demand policy of an agent could depend on the agent's utility function, on its stocks, on global prices and, possibly, on other model-specific parameters. However, it would not depend on, e.g., the stocks of other agents.

In the case of the trade-resolving policy, the action is an elementary bilateral exchange, the outcome of the trade. The state consist of the offers and demands of the interacting agents, say a_1 and a_2, and on model-specific parameters. We say that a trade-resolving policy is agent-independent if it does not depend explicitly on a_1, a_2.[10]

Informally, one can describe an elementary bilateral trade between two agents a_1, a_2 as a three-step process. In the first step, the two agents apply their offer and demand policies. This yields (o_1,g_1), (d_1,g_1'), (o_2,g_2) and (d_2,g_2') in $\mathbb{R}_{\geqslant 0} \times G$ with the following interpretation: o_1 is the amount of g_1 that a_1 offers in exchange for d_1 units of g_1'. Similarly, o_2, d_2 are the amounts

[9] The offer and demand policy has not to be confused with the demand function discussed in Section 3.2. The case in which an agent's offer and demand policy depends on its demand function is, however, an important one, see Section 4.2.

[10] Of course, trade-resolving policies do usually depend on the interacting agents implicitly through their offers and demands. However, they do not depend on the agents "identities": every pair of agents is treated as every other pair. One could imagine models of exchange with agent-dependent trade-resolving policies, e.g., to model "privileges".

g_2, g_2' offered and asked by a_2. A minimal specification for the outcome (o,g), (d,g') of the offer and demand policy of an agent a is:

$$x\ a\ g - o \geqslant 0, \tag{13}$$

where $x\ a \in Q$ are the stocks of a. Eq. (13) requires the amount of g offered by a to be "in budget". The idea is that if the agent's offer and demand are matched (by another agent's demand and offer) at most o units of g will be withdrawn from its stock[11]: the specification ensures that the result will be non-negative (remember that $Q = G \to \mathbb{R}_{\geqslant 0}$).

In the second step, the trade-resolving policy is applied to the offer and demands (o_1, g_1), (d_1, g_1'), (o_2, g_2) and (d_2, g_2') of a_1 and a_2. The result are two pairs (δ_1, g_1) and (δ_2, g_2) in $\mathbb{R}_{\geqslant 0} \times G$, the amounts of goods to be exchanged. Minimal requirements for the trade-resolving rule are:

$$\begin{aligned} &g_1 \neq g_2' \ \vee\ g_1' \neq g_2 \ \Rightarrow\ \delta_1 = \delta_2 = 0, \\ &g_1 = g_2' \ \wedge\ g_1' = g_2 \ \wedge\ d_1 \leqslant o_2 \ \wedge\ d_2 \leqslant o_1 \ \Rightarrow\ \delta_1 = d_1 \ \wedge\ \delta_2 = d_2. \end{aligned} \tag{14}$$

The interpretation of the above specification is: for a non-zero exchange to take place, the goods demanded (offered) by a_1 have to coincide with the goods offered (demanded) by a_2. If this condition is fulfilled and the demands of a_1, a_2 are both matched by the corresponding offers, then the outcome of the trade – the amounts of g_1, g_2 to be exchanged, shall coincide with the demands of the agents. One can imagine a number of resolving rules which match the above specification. A simple "fill or kill" one, for instance, could return zero exchanges whenever the left-hand side of Eq. (14) evaluates to false. We discuss a more complicated rule in [6] where we give a full specification of the bilateral trading mechanism implemented in [19].

In the last step of an elementary bilateral trade between a_1 and a_2, the two agents actually exchange δ_1 units of g_2 and δ_2 units of g_1. The result is an allocation x' such that

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

$$(x'\ a_1 - x\ a_1)\ g_2 = \delta_1, \tag{15}$$

$$(x'\ a_2 - x\ a_2)\ g_1 = \delta_2. \tag{16}$$

If the goods g_1, g_2 offered by a_1, a_2 are given by a function $s \in A \to G$, see Section 3.5, one can formulate the effect of an elementary bilateral trade on allocations through a function

$$\begin{aligned} &ebt \in (A \to Q) \to A \times A \to (A \to Q), \\ &x' = ebt\ x\ (a_1, a_2) \Rightarrow x(\mathcal{X}_e\ a_1\ a_2\ (s\ a_2)\ (s\ a_1))x'. \end{aligned} \tag{17}$$

The requirements (15), (16) on x' imply that ebt is non-decreasing in the demand good and non-increasing in the offer good:

$$x' = ebt\ x\ (a_1, a_2) \ \wedge\ g_1 = s\ a_1 \ \wedge\ g_2 = s\ a_2$$

$$\Rightarrow$$

$$x'\ a_1\ g_2 \geqslant x\ a_1\ g_2 \ \wedge\ x'\ a_1\ g_1 \leqslant x\ a_1\ g_1 \tag{18}$$

$$\wedge$$

$$x'\ a_2\ g_1 \geqslant x\ a_2\ g_1 \wedge x'\ a_2\ g_2 \leqslant x\ a_2\ g_2. \tag{19}$$

In theoretical and computational economics, the relationship between the notion of equilibrium and models of exchange is a subject of ongoing research [19,20,36,40]. In this context, a natural question is whether it is possible to identify economically sound trading policies – agent-specific offer and demand policies and agent-independent trade-resolving policies – which realize equilibrium allocations through sequences of elementary bilateral trades.

The question is interesting because attempts at explaining equilibrium allocations and prices as stationary states of sequences of bilateral interactions between agents have not, so far, been convincing. In particular, we do not know economically plausible bilateral trade mechanisms which provably lead (arbitrarily near) to equilibrium prices and allocations in a finite number of interactions and for all initial allocations and utility profiles that guarantee existence of Walrasian equilibria.

The question of whether it is possible to identify economically sound trading policies which realize equilibrium allocations through sequences of elementary bilateral trades can only be studied in a well-defined context. This has to specify the offer and demand policies of the agents, the trade-resolving policy and the trading sequences that is, which agent pairs are interacting and the order of interaction. In the following sections we introduce the elementary notions which are needed to formulate such context.

[11] And, of course, added to the stock of g of the other agent.

3.5. Sectors, trading schedules, trading rounds

In agent-based economic modeling, the notion of sector is not clearly defined. Sometimes sectors are used, informally, to denote different kinds or types of agents: households, firms, banks. In this context, the notion of agent itself is often taken as primitive: what agents of different sectors have in common is not specified.

Sometimes sectors are used to denote certain relationships between agents of different kinds. For instance, households might be linked to banks through certain relations "in the financial sector" and to firms (through other relations) "in the production sector".

In models of exchange, sectors are often used to organize the interactions between agents [19,20]. When these interactions are elementary bilateral trades, sectors are defined in terms of goods. All agents of a given sector have in common a certain sector-specific good. Different sectors are characterized by different goods. In these models, sectors are represented by a surjective function

$$\begin{aligned} &s \in A \to G, \\ &\forall\, g \in G, \quad s^{-1}\ g \neq \emptyset. \end{aligned} \tag{20}$$

In Eq. (20), $s^{-1}\ g$ denotes the inverse image of g through s: $s^{-1}\ g = \{a \in A \mid s\ a = g\}$. In models of exchange which are based on evolutionary dynamics [11,19,20], sectors are often equally populated and $x_0 \in A \to Q$ and $u \in A \to U$ are sector-wise constant:

$$\forall\, g, g' \in G, \quad |s^{-1}\ g| = |s^{-1}\ g'|, \tag{21}$$

$$\forall\, a, a' \in A, \quad s\ a = s\ a' \ \Rightarrow\ x_0\ a = x_0\ a' \ \wedge\ u\ a = u\ a'. \tag{22}$$

In models which fulfill (21), $|A|$ is a multiple of $|G|$ and the number of agents per sector is $|A|/|G|$. In models of exchange based on elementary bilateral trades between sectors, interactions are often organized in *trading rounds*. In a round, all agents of every sector interact with a certain number of agents of every other sector. This number is given by a "number of peers" function *np*. A minimal specification for *np* is

$$\begin{aligned} &np \in G \to G \to \mathbb{N}, \\ &\forall\, g, g' \in G, \quad np\ g\ g' \leqslant |s^{-1}\ g'|. \end{aligned} \tag{23}$$

The sequence of interactions which take place in a round is represented by a trading schedule *ts*. This is a list[12] of agent pairs: $ts \in List\ (A \times A)$. A minimal specification for *ts* is:

$$\forall\, a \in A, \quad \forall\, g \in G, \quad |\{(a, a') \mid \text{elem}\ (a, a')\ ts, \quad s\ a' = g\}| = \begin{cases} np\ (s\ a)\ g & \text{if } g \neq s\ a, \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

The schedule *ts* specifies both the interactions which take place in a trading round and the order in which they take place. In general, the outcome of a trading round depends on this order.[13] Thus, a trading schedule effectively defines a network of agents. When $np\ g\ g'$ is bounded independently of the number of agents in $s^{-1}\ g'$, the network is sometimes referred to as a *social* network. Often, the $np\ g\ g'$ agents of sector g' which are trading with a given agent of sector $g \neq g'$ are drawn randomly in $s^{-1}\ g'$. In this case the trading schedule is called a random schedule or a random network.

In a trading round, the pairs of *ts* engage in elementary bilateral trades one after the other. The effect of a trading round with schedule *ts* on an allocation x is expressed by folding *ebt* on *ts* from x:

$$\text{fold}\ ebt\ x\ ts. \tag{25}$$

Like elem, fold is a *polymorphic* function. Its type depends on two parameters X and Y:

$$\text{fold} \in (X \to Y \to X) \to X \to List\ Y \to X. \tag{26}$$

In a trading round (25), $X = A \to Q$ and $Y = A \times A$ according to the specification of *ebt* (17). Like many functions that operate with lists, see [5], fold is defined recursively by *pattern matching* the case in which the list argument is empty

$$\text{fold}\ f\ x\ [] = x$$

[12] Lists are recursively defined data types [5]. We denote an empty list with the symbol []. A list consisting of a $x \in X$ put on the top of a list $xs \in List\ X$ is denoted by $(x : xs)$. We say that $xs!!k$ is the kth element of xs. The boolean function

$$\begin{aligned} &\text{elem} \in X \to List\ X \to Bool, \\ &\text{elem}\ x\ xs = \begin{cases} \text{true} & \text{if } \exists\, k \in \mathbb{N} \text{ such that } xs!!k = x, \\ \text{false} & \text{otherwise} \end{cases} \end{aligned}$$

is used to test list membership.

[13] This is the main reason for representing trading schedules in terms of lists. If the order of trading would not matter, one could simply use sets of agent pairs.

and the case in which the list argument consists of a $y \in Y$ on the top of a (possibly empty) list $ys \in List\ Y$:

$$\text{fold}\ f\ x\ (y : ys) = f\ (\text{fold}\ f\ x\ ys)\ y.$$

As an example, consider the effect on x of folding f on $[(a_1,a_2), (a_3,a_1), (a_2,a_3)]$[14]:

$$\begin{aligned}
&\text{fold}\ f\ x\ [(a_1,a_2),(a_3,a_1),(a_2,a_3)]\\
&=\\
&\text{fold}\ f\ x\ ((a_1,a_2)\ :\ [(a_3,a_1),(a_2,a_3)])\\
&=\\
&f\ (\text{fold}\ f\ x\ ((a_3,a_1)\ :\ [(a_2,a_3)]))\ (a_1,a_2)\\
&=\\
&f\ (f\ (\text{fold}\ f\ x\ ((a_2,a_3)\ :\ []))\ (a_3,a_1))\ (a_1,a_2)\\
&=\\
&f\ (f\ (f\ (\text{fold}\ f\ x\ [])\ (a_2,a_3))\ (a_3,a_1))\ (a_1,a_2)\\
&=\\
&f\ (f\ (f\ x\ (a_2,a_3))\ (a_3,a_1))\ (a_1,a_2).
\end{aligned}$$

Notice that the trading schedule $[(a_1,a_2), (a_3,a_1), (a_2,a_3)]$ is "executed" from the right to the left. First, one accounts for the bilateral trade between a_2 and a_3 on x. This results in a new allocation $x' \in X$. Then, the elementary trade between a_3 and a_1 yields another allocation x''. Finally, the interaction between a_1 and a_2 completes the round. For this simple example it is easy to rewrite the computation in a more explicit form:

$$\begin{aligned}
&\text{fold}\ f\ x\ [(a_1,a_2),(a_3,a_1),(a_2,a_3)] = x'''\\
&\text{where}: x' = f\ x\ (a_2,a_3),\\
&\qquad x'' = f\ x'\ (a_3,a_1),\\
&\qquad x''' = f\ x''\ (a_1,a_2).
\end{aligned}$$

We conclude this section with two remarks. First, trading rounds inherit the properties of elementary bilateral trades: if $x' = \text{fold}\ ebt\ x\ ts$, then x' is a re-allocation of x. Moreover, for any $a \in A$, $x'\ a\ (s\ a) \leqslant x\ a\ (s\ a)$ and, for any $g \in G$, $g \neq s\ a \Rightarrow x'\ a\ g \geqslant x\ a\ g$. If ebt is value-preserving, trading rounds obtained by folding ebt are value-preserving.

Second, a trade schedule can be seen as a relation between agents together with a "time" plan. The relation defines a network of interactions. The time plan their sequence. Within this interpretation, however, the notion of time is ambiguous. One can think of a trading round as a process in which exactly one trade takes place at every time step. In this case, the number of time steps of a trading round would be given by the length of ts. However, one can also think of ts as the flattening of a list of lists of agent pairs, every sublist consisting of interactions which can take place simultaneously without changing the result of the round. With this interpretation, the number of time steps would be given by the number of sublists in which ts can be decomposed.[15]

3.6. Trading games, dynamical models of exchange

Trading games are games built-up of trading rounds. As explained in the previous section, in a trading round agent pairs engage in a number of elementary bilateral trades starting from a given initial allocation. The result is a new allocation.

In a trading game, each round results in a new allocation and in an outcome – the outcome of that round. The outcome is game-specific. In a game designed to measure the effectiveness of the agents offer and demand policies, for instance, the outcome could be a trading fitness function, perhaps a (possibly discounted) sum of the utilities of the allocations achieved by the agents in the round. The result of the whole game depends on the allocations and on the outcomes obtained in a sequence of rounds.

Depending on the assumptions made on the initial data of the single rounds, one can formulate different kinds of games. A simple one is one in which the initial data are the same in each round. In this case, one can formulate the game in terms of a sequence of trading schedules – a list of lists of agent pairs $tss \in List\ (List\ (A \times A))$ – and of two functions: an extended elementary bilateral trade function

$$\begin{aligned}
&eebt \in O \times (A \to Q) \to A \times A \to O \times (A \to Q),\\
&(o', x') = \text{fold}\ eebt\ (o,x)\ ts\ \Rightarrow\ x' = \text{fold}\ ebt\ x\ ts
\end{aligned} \tag{27}$$

[14] For $A = \{a_1,a_2,a_3\}$, $G = \{g_1,g_2,g_3\}$ and $s\ a_k = g_k$, $k \in \{1,2,3\}$, $np\ g\ g' = 1$ and $ts = [(a_1,a_2), (a_3,a_1), (a_2,a_3)]$ fulfill (23) and (24). Thus, they are valid "number of peers" and "trading schedule function", respectively.

[15] Finding such sublists is an interesting problem. Solving this problem is a pre-condition for accelerating the computation of trading rounds through concurrency.

and a function $h \in List\ (O \times (A \to Q)) \to R$ which computes the result of the whole game

$$h\ (\text{map}\ (\text{fold}\ eebt\ (o_0, x_0)\ )\ tss). \tag{28}$$

In the above equation, (o_0, x_0) is the initial data. In the kth round (the number of rounds is given by the number of schedules of tss), $eebt$ is folded on $tss!!k$ starting from (o_0, x_0). The initial data are the same in all rounds. The result of the rounds is a list of pairs output-allocation. The result of the whole game is computed by applying h to this list. The specification (27) requires the action of $eebt$ on allocations to be an elementary bilateral trade. The nature of O and the action of $eebt$ on $o \in O$ depends on the particular model considered.

Dynamical models of exchange are models of exchange in which a number of functions, e.g., prices or allocations, explicitly depend on time. In this paper, we only consider time-discrete models. For such models, functions which explicitly depend on time can be formulated as functions on natural numbers. Time-dependent allocations, for instance, can be formulated as functions of type $\mathbb{N} \to A \to Q$. We write (slightly abusing the notation introduced in Section 2)

$$x \in \mathbb{N} \to A \to Q$$

to posit that x is a time-dependent allocation. In dynamical models of exchange, it is important to distinguish between functions that depend on time explicitly and functions that depend on time implicitly through other functions. For a given model, we denote the set of all functions which depend on time explicitly by Z.

For instance, in a model in which only allocations depend on time explicitly, $Z = A \to Q$. In a model in which allocations and prices depend on time explicitly $Z = (A \to Q) \times P$.

Thus, in general, we can represent the model functions that depend on time explicitly by means of a function z of type $\mathbb{N} \to Z$. In the case of time-dependent allocations $x \in \mathbb{N} \to A \to Q$ and prices $p \in \mathbb{N} \to P$, for instance, z is defined by

$$\begin{aligned} &z \in \mathbb{N} \to (A \to Q) \times P, \\ &z\ t = (x\ t, p\ t). \end{aligned}$$

Thus, dynamical models of exchange can be viewed as dynamical systems and the value of z at time t, $z\ t$ is often called the *state* (of the system, model) at t. The difference $z\ (t+1) - z\ t$ is called the state *transition* at t. We are interested in dynamical models of exchange in which

- The state at $t = 0$, $z\ 0$, is given.
- The states $z\ 1$, $z\ 2$, ... are computed from $z\ 0$ by iterating a transition function tr.
- The transition function takes as arguments a state (the "current" state) and a random variable[16]:

$$tr \in Z \to \Omega \to Z.$$

In other words, we are interested in dynamical models of exchange in which $z\ 0, z\ 1, z\ 2, \ldots$ are given by

$$\begin{aligned} &z\ 0 = z_0, \\ &z\ (t+1) = tr\ (z\ t)\ (\omega\ t). \end{aligned} \tag{29}$$

In [6] we show that the model presented in [19] is a dynamical model of exchange in the sense of (29). In this model $z \in \mathbb{N} \to A \to P$ represents time-dependent, agent-specific "private" prices and tr is an evolutionary algorithm obtained by folding a copy-mutate rule on $\omega\ t$:

$$tr\ (z\ t)\ (\omega\ t) = \text{fold}\ cm\ (z\ t)\ (\omega\ t).$$

The random variables $\omega\ 0$, $\omega\ 1$, ... are schedules similar to those of Section 3.5. The copy-mutate rule cm is specified in terms of a "trading fitness" function $f \in A \to \mathbb{R}$.

This is obtained, at each iteration step of (29), by playing a simple trading game of the kind described by equations (27) and (28). In such game the initial data are the same in all rounds and the output of a single round o' is the trading fitness achieved by the agents in that round. The tth trading game is played at fixed (private) prices $z\,t$. These control both the offer and demand policies of the agents and the trade resolving policy embedded in $eebt$.

4. Dynamical models of exchange and equilibrium

In the last section we have presented a framework for agent based models of exchange. The framework can be applied in a number of ways. In this section we discuss three ones.

[16] Or, from the dynamical systems viewpoint, the system is non-deterministic, explicit and autonomous.

4.1. Model implementation

In implementing models of exchange, e.g., in an imperative programming language with limited support for constrained polymorphism like C++, one can use the specifications derived in Section 3 directly to formulate pre- and post-conditions for specific function interfaces.

As an example, let `ebt` be an interface for implementing elementary bilateral trades between pairs of agents in the context of a model of exchange in which trading takes place between sectors, see Section 3.5. Let agents, goods and sectors be represented by values of type `Agent`, `Good` and `Sector`, respectively and let allocations be represented by values of type `Allocation`.[17] Further, assume that `Sector` is a type synonym of `Good` and that `ebt` can rely on a `sector` function and on a `is_elementary_bilateral_exchange` function

```
Sector sector (const Agent& a);
Bool is_elementary_bilateral_exchange (const Allocation& x,
                                       const Allocation& xp,
                                       const Agent& a1,
                                       const Agent& a2,
                                       const Good& g1,
                                       const Good& g2);
```

implementing specifications (20) and (11), (12). Then, in a design-by-contract approach [30], an interface for `ebt` consistent with specification (17) could look like

```
void ebt(Allocation& x, const Agent& a1, const Agent& a2) {
  require(sector(a1) != sector(a2));
  let(Allocation y = x);
  let(Good g1 = sector(a1));
  let(Good g2 = sector(a2));
  ...
  ensure(is_elementary_bilateral_exchange (x,y,a1,a2,g1,g2));
  ensure(x(a1)(g2) >= y(a1)(g2));
  ensure(x(a2)(g1) >= y(a2)(g1));
}
```

Of course, pre- and post-conditions cannot guarantee, in general, implementations to be correct, that is, to fulfill their specifications. In practice, however, systematic usage of design-by-contract constructs has turned out to be extremely useful, especially in software prototyping and re-factoring.

As a bonus, pre- and post-conditions expressed through domain-specific interfaces provide a straightforward form of model documentation.

4.2. Model analysis and problem formulation

In Section 3.4 we raised the question whether it is possible to find economically sound trading policies which realize equilibrium allocations through sequences of elementary bilateral trades.

With the framework of Section 3 in place, we are now ready to formulate such question precisely. To this end, consider a dynamical model of exchange (29) in which only allocations depend on time explicitly

$$\begin{aligned} &x\ 0 = x_0, \\ &x\ (t+1) = tr\ (x\ t)\ (\omega\ t) \end{aligned} \tag{30}$$

and x_0, tr and $\omega\ t$, $t = 0, 1, \ldots$ are defined in terms of a sector function $s \in A \to G$ (specification (20)), of a sector-to-sector number of peers function $np \in G \to G \to \mathbb{N}$ (specification (23)) and of a utility profile $u \in A \to U$.

We assume, for simplicity, that sectors are equally populated (21) and that the number of peers is constant

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

We also assume the iteration (30) to start from a sector-wise constant initial allocation which is different from zero only in the sector-specific good

[17] In a simple setup, `Agent`, `Good` and `Sector` could just be type synonyms of `Nat`, a type representing a fixed zero-based range of natural numbers, e.g., up to 2^{32}. `Allocation` could be a type synonym of `Array < Array < Real > >` and `Array` could be a type suitably derived from the STL (Standard Template Library, www.sgi.com/tech/stl) class `vector`.

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

and x_0, u to be such that the optimality equation (4) admit unique solutions for arbitrarily prices. In other words, we assume the simple setup discussed in Section 3.2 in which (x_0, u) admit a demand profile d.

Consider a simple transition function in which $\omega\ t$, $t = 0, 1, \ldots$ are random trading schedules (specification (24)) and $x\ (t+1)$ is computed from $x\ t$ in a single round of elementary bilateral trades

$$tr\ (x\ t)\ (\omega\ t) = \text{fold}\ ebt\ (x\ t)\ (\omega\ t).$$

In economic terms, this setup can be viewed as modeling a situation in which, at time $t = 0$, all agents have terminated a production step in a given sector and are equipped with a certain amount of their production good (in fact, the same amount of their sector-specific good for all agents of a given sector). At each step of the iteration, all agents of all sectors engage with n_p agents of every other sector in elementary bilateral trades. The sequence of interactions is controlled by the random schedule of that step and the idea is that agents attempt, via trading under given equilibrium prices, to satisfy their excess demand and achieve their optimal allocations.[18]

As explained in Section 3.4, *ebt* (and, therefore, *tr*) is defined, in general, in terms of agent-specific offer and demand policies and of an agent-independent trade-resolving policy. Let us call these policies *odp* and *trp*, respectively. We are particularly interested in the case in which the offer and demand policies are parameterized on prices, *ebt* is non-decreasing in the demand good and non-increasing in the offer good and all agents act according to the same (offer and demand) policy.[19] In the notation introduced in Section 3.4, we can express these assumptions on *ebt* through the following specification:

$$\begin{aligned}
&trp \in P \to \mathbb{R}_{\geqslant 0} \times G \to \mathbb{R}_{\geqslant 0} \times G \to \mathbb{R}_{\geqslant 0} \times G \to \mathbb{R}_{\geqslant 0} \times \mathbb{R}_{\geqslant 0}\\
&odp \in P \to Q \to G \to G \to \mathbb{R}_{\geqslant 0} \times \mathbb{R}_{\geqslant 0}\\
&ebt\ x_t\ (a_1, a_2) = x'_t\\
&\quad \Rightarrow\\
&x_t(\mathcal{X}_e\ a_1\ a_2\ g_2\ g_1)x'_t\ \wedge\ (x'_t\ a_1 - x_t\ a_1)\ g_2 = \delta_1\ \wedge\ (x'_t\ a_2 - x_t\ a_2)\ g_1 = \delta_2\\
&\text{where}:\ (\delta_1, \delta_2) = trp\ p\ (o_1, g_1)\ (d_1, g_2)\ (o_2, g_2)\ (d_2, g_1),\\
&\qquad\quad (o_1, d_1) = odp\ p\ (x_t\ a_1)\ g_1\ g_2,\\
&\qquad\quad (o_2, d_2) = odp\ p\ (x_t\ a_2)\ g_2\ g_1,\\
&\qquad\quad g_1 = s\ a_1,\\
&\qquad\quad g_2 = s\ a_2.
\end{aligned}$$

To formulate the problem of finding economically sound trading policies which realize equilibrium allocations through sequences of elementary bilateral trades we still have to express what it means for trading policies to be "economically sound".

The problem has two sides. The first one – what it means for a trading policy to be economically sound – is a modeling issue we are not concerned with here (but see our remarks on the criticisms of the notions of individual rationality in the introduction).

The second side of the problem is *how* to formulate whatever it means for a trading policy to be economically sound into specifications for *odp* and *trp*. Here is an example of one such formulations. Suppose, as in [19], that economically sound trading policies are to

(a) be demand-driven.
(b) yield elementary bilateral exchanges which, for given prices, do not decrease the values (according to those prices) of the stocks of the interacting agents.

We can formulate (a) by requiring *odp* to fulfill

$$d\ x_0\ u\ a\ p\ g' - x_t\ a\ g' = 0\ \wedge\ (o, d) = odp\ p\ (x_t\ a)\ g\ g' \ \Rightarrow\ d = 0.$$

Then (b) implies that o must be zero as well[20]:

$$d\ x_0\ u\ a\ p\ g' - x_t\ a\ g' = 0\ \Rightarrow\ odp\ p\ (x_t\ a)\ g\ g' = (0, 0). \tag{31}$$

With the above definitions and specifications for x_0, x, tr, ω, np, s, *ebt*, *odp* and *trp* in place, the problem of finding sound trading policies which realize equilibrium allocations through sequences of elementary bilateral trades can be finally formulated

[18] In the economic interpretation, the underlying goal of trading – achievement of optimal allocations – is often "explained" by a subsequent step in which the agents are thought to consume their goods thereby receiving benefits proportional to the achieved utility.

[19] In [6] we show that the elementary bilateral trades at the core of the model presented in [19] enjoy these properties.

[20] Remember that *odp* yields upper bounds for the quantities of the "own" goods to be exchanged in an interaction. If a's offer and demand (o, d) of (g, g') are matched by another agent offer and demand (o', d') (of (g', g)), at most o units of g will be withdrawn from its stock. A zero demand is obviously matched by any o'. Thus, if o was greater than zero, any pair (o', d') with $d' \leqslant o$ would match $(o, 0)$, $trp\ p\ (o, g)\ (0, g')\ (o', g')(d', g)$ would yield $(0, d')$ (Eq. (14)) and d' units of g would be withdrawn from the stock of a without compensation. This contradicts (b) and therefore o has to be zero.

as the problem of finding *odp* and *trp* that guarantee (30) to converge, in a finite number of steps, towards x_e for all x_e, p, x_0, u that fulfill $(x_e, p)\mathcal{E}(x_0, u)$. Specification (31) is obviously sufficient to guarantee that equilibrium allocations are fixed-points of (30) that is

$$(x_e, p)\mathcal{E}(x_0, u) \;\Rightarrow\; x_e = \text{fold } ebt\ x_e\ (\omega\ t). \tag{32}$$

Unfortunately, it is not strong enough to guarantee that all fixed-points of (30) are equilibrium allocations, not even for the simple case in which u is constant.

We conclude this section by remarking that the trading policies proposed in [19] do not fulfill specification (14). However, they satisfy (31) and fulfill two additional requirements. They are value-preserving and demand-limited:

$$\begin{aligned}&(\delta_1, \delta_2) = trp\ p\ (o_1, g_1)\ (d_1, g_2)\ (o_2, g_2)\ (d_2, g_1)\\ &\quad\Rightarrow\\ &\delta_1 * (p\ g_2) = \delta_2 * (p\ g_1)\ \wedge\ (\delta_1 \leqslant d_1\ \wedge\ \delta_2 \leqslant d_2)\end{aligned}$$

Value preserving trade-resolving policies guarantee that if (x_e, p) is in equilibrium with (x_0, u) then it is in equilibrium with $((x\ t), u)$ for all times t.[21] Value-preserving, demand-limited trade-resolving policies guarantee convergence of (30) (towards equilibrium allocations) for some special, constant utility profile. The results discussed in [6], however, suggest that the requirement are not strong enough to grant convergence when, e.g., the utility profile is sector-wise constant but different from sector to sector.

4.3. Model analysis and validation

Once a model has been specified, one can start reasoning about the model[22] rigorously. In particular, one can deduce properties that model implementations which are consistent with the specification have to exhibit.

This immediately allows one to design *crucial* experiments for validating model implementations. Crucial experiments are experiments for which a negative outcome unambiguously indicates errors in the implementation.

For example, for the (class of) dynamical models of exchange outlined in the previous section, one can deduce that the total excess demand

$$\begin{aligned}&td\ u\ p\ x \in \mathbb{N} \to \mathbb{R}_{\geqslant 0},\\ &td\ u\ p\ x\ t = \sum_{a \in A} \sum_{g \in G} (d\ (x\ t)\ u\ a\ p\ g - (x\ t)\ a\ g)\end{aligned}$$

of the time-dependent allocations x is, for value-preserving trade-resolving policies, non-increasing in time. Therefore, a model implementation yielding allocations whose total excess demand at some time t^* happens to be smaller than at some later time cannot be correct.

When model implementations cannot be automatically derived from specifications and one cannot prove model implementations to be correct (to comply with the specifications), crucial experiments and corner-case model setups[23] play an essential role in validating model implementations.

In this case, validating a model implementation simply means assessing that the implementation does not fail in a number of crucial experiments and corner-case studies. Of course, such assessment can only be done for a finite number of cases and cannot not show that the implementation is correct.

However, systematically exercising a model implementation on carefully chosen sets of crucial experiments is, in absence of rigorous proofs of correctness, the best one can do when unambiguous model specifications are available.

When not, very little can be said about a model. In this case prototype implementations can still be used to compute numerical results. However, very little can be said about the correctness of such implementations and interpretations of the numerical results become problematic.

5. Conclusions

We have presented a functional framework for describing dynamical models of exchange economies. The framework consists of a notation and of a number of elementary notions. The notation is a blend between standard mathematical notation and simple constructs of typed functional programming languages.

[21] Remember that *tr* inherits the properties of *ebt* and that value-preserving reallocations preserve equilibrium, see Section 3.1.

[22] A model specification is almost never a model definition. Thus, properly speaking, model specifications allow one to reason about properties of models classes, not just single models.

[23] Corner-case model setups – corner cases in short – are model setups for which it is particular easy to reason about the model and deduce properties that correct model implementations have to satisfy. Corner cases are often obtained by selecting limit values for specific model parameters. For the dynamical models of exchange outlined in the previous section, for instance, typical corner-cases can be obtained by considering two goods or one single agent per sector.

Some of the specifications introduced in Section 3 can be directly translated into Haskell data types or type classes. A translation of the whole framework into an executable programming language, however, would require a dependently typed language and go well beyond the scope of this article.

As explained in the introduction, the framework has been motivated by the attempt at re-implementing the model described in [19] and has been conceived to support the specification of an interesting but narrow class of models of exchange. For models of exchange based on, e.g., multilateral interactions between agents, non-deterministic or stochastic transition functions, reinforcement learning, the framework needs to be generalized and extended.

In spite of these shortcomings, the proposed framework is, at the best of our knowledge, the first attempt at introducing (in our view much needed) mathematical specifications in computational economics, in particular for agent-based models of exchange.

In the last section, we have shown that such specifications (1) effectively support disciplined model implementation, e.g., by providing pre- and post-conditions and crucial experiments for model validation; (2) assist the analysis of computational models and allow one to precisely formulate model-specific questions, e.g., for the dynamical models of exchange outlined in Section 4.2, about the asymptotic behavior of time-dependent allocations and equilibrium allocations.

We have applied the functional framework for the complete specification and for the implementation of the model presented in [19]. This application goes beyond the scope of this work and is presented in a companion paper [6].

Acknowledgements

The authors thank the reviewers and the editor, whose comments have triggered various nontrivial improvements of the paper. It is our pleasure to thank R. Klein[24]: his remarks and suggestions have lead to significant improvements of a first draft of this 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.

Appendix A

We have to show that, with the utility profile (6), (7) and positive prices $p \in P$, $x \in A \to Q$ as defined by Eq. (8) satisfies (3) and (4). As far as (3) is concerned, one has:

$$\forall\, g \in G$$

$$\sum_{a \in A} x\ a\ g$$

$$= \{\text{equation } (8)\}$$

$$\sum_{a \in A} \frac{(x_0\ a) \cdot p}{w \cdot p} * (w\ g)$$

$$= \{\text{utility weights collinearity } (7)\}$$

$$\frac{1}{\lambda} * \frac{w \cdot p}{w \cdot p} * (w\ g)$$

$$= \{\text{utility weights collinearity, again } (7)\}$$

$$\frac{1}{\lambda} * \lambda * \left(\left(\sum_{a \in A} x_0\ a \right) g \right)$$

$$= \{\text{finite sum of functions}\}$$

$$\sum_{a \in A} x_0\ a\ g.$$

Hence, x satisfies (3). Let us now show that x satisfies (4). First of all, one has

$$\forall\, a \in A, \quad (x\ a) \cdot p = (x_0\ a) \cdot p,$$

because of Eq. (8). Thus, for every a in A, x satisfies the constraint of Eq. (4). Assume there exists an $x'\ a \in Q$ yielding a higher utility than $x\,a$. Then

[24] Applied mathematics, Geophysical Fluid Dynamics. Free University, Berlin.

$$
\begin{aligned}
&(u\ a)\ (x'\ a) > (u\ a)\ (x\ a)\\
&= \{\text{equation } (6)\}\\
&\min_{g\in G}((x'\ a)\ g)\ /\ (w\ g) > \min_{g\in G}((x\ a)\ g)\ /\ (w\ g)\\
&= \{\text{equation } (8)\}\\
&\min_{g\in G}((x'\ a)\ g)\ /\ (w\ g) > \frac{(x_0\ a)\cdot p}{w\cdot p}\\
&=\\
&\forall\ g\in G,\quad ((x'\ a)\ g)\ /\ (w\ g) > \frac{(x_0\ a)\cdot p}{w\cdot p}\\
&=\\
&\forall\ g\in G,\quad (x'\ a)\ g > \frac{(x_0\ a)\cdot p}{w\cdot p}(w\ g)\\
&\Rightarrow\\
&(x'\ a)\cdot p > (x_0\ a)\cdot p,
\end{aligned}
$$

that is, x' violates the budget constraint of Eq. (4). Thus, there is no x' better than x which satisfies the budget constraint. Thus, x satisfies (4).

References

[1] J. Abrial, The B-Book: Assigning Programs to Meanings, Cambridge University Press, 1996.
[2] A. Aho, J. Ullman, J. Hopcroft, Data Structures and Algorithms, Addison-Wesley, 1982.
[3] W.B. Arthur, V. Morrison, S.N. Durlauf, (Eds.), The Economy as an Evolving Complex System II. Santa Fe Institute Studies in the Sciences of Complexity Lecture Notes, 1997.
[4] R. Backhouse, Program Construction: Calculating Implementations from Specifications, John Wiley & Sons, 2003.
[5] R. Bird, Introduction to Functional Programming using Haskell, second ed., International Series in Computer Science, Prentice Hall, 1998.
[6] N. Botta, A. Mandel, M. Hofmann, S. Schupp, Mathematical specifications for agent-based models of exchange: a case study, Applied Mathematics and Computation, submitted for publication.
[7] J. Bowen, Formal Specification and Documentation using Z: A Case Study Approach, International Thomson Computer Press, 1996.
[8] M. Clavel, F. Durn, S. Eker, P. Lincoln, N. Mart-Oliet, J. Meseguer, C. Talcott, All About Maude – A High-Performance Logical Framework, Lecture Notes in Computer Science, Springer-Verlag, 2007.
[9] M. Cohn, User Stories Applied: For Agile Software Development, Addison-Wesley Signature Series, Addison-Wesley, 2004.
[10] D. Colander, F. Föllmer, H. Haas, M. Goldberg, M. Juselius, K. Kirman, T. Lux, B. Sloth, The Financial Crisis and the Systemic Failure of Academic Economics. Technical report, Kiel Working Paper 1489, Kiel Institute for the World Economy, 2009.
[11] H. Dawid, Evolutionary game dynamics and the analysis of agent-based imitation models: the long run, the medium run and the importance of a global analysis, Journal of Economic Dynamics and Control 31 (2007) 2108–2133.
[12] G. Debreu, Theory of Value: An Axiomatic Analysis of Economic Equilibrium, Cowles Foundation Monographs Series, Yale University Press, 1959.
[13] S.J. DeCanio, Economic Models of Climate Change: A Critique, Palgrave Macmillan, New York, 2003.
[14] E.W. Dijkstra, A Discipline of Programming, Prentice-Hall Series in Automatic Computation, Prentice-Hall, 1976.
[15] D. Colander (Ed.), Post Walrasian Macroeconomics, Cambridge University Press, 2006.
[16] S.L. Peyton Jones et al, Haskell 98 Language and Libraries: the Revised Report, Cambridge University Press, 2003.
[17] M. Fowler, UML Distilled, third ed., Addison-Wesley Object Technology Series, Addison-Wesley, 2003.
[18] M. Gallegati, M. Richiardi, Agent Based Models in Economics and Complexity, in: R.A. Meyers (Ed.), Encyclopedia of Complexity and Systems Science, Springer, 2009, pp. 200–223.
[19] H. Gintis, The emergence of a price system from decentralized bilateral exchange, B.E. Journal of Theoretical Economics 6 (2006) 1302–1322.
[20] H. Gintis, The dynamics of general equilibrium, Economic Journal 17 (2007) 1280–1309.
[21] V. Grimm, U. Berger, F. Bastiansen, S. Eliassen, V. Ginot, J. Giske, J. Goss-Custard, T. Grand, S.K. Heinz, G. Huse, A. Huth, J.U. Jepsen, C. Jorgensen, W.M. Mooij, B. Muller, G. Pe'er, C. Piou, S.F. Railsback, A.M. Robbins, M.M. Robbins, E. Rossmanith, N. Ruger, E. Strand, S. Souissi, R.A. Stillman, R. Vabo, U. Visser, D.L. Deangelis, A standard protocol for describing individual-based and agent-based models, Ecological Modelling 198 (1–2) (2006) 115–126.
[22] A. Simon Herbert, Models of Man, Wiley, New York, 1957.
[23] C.B. Jones, Systematic Software Development Using VDM, second ed., Prentice-Hall International Series in Computer Science, Prentice-Hall, 1990.
[24] D. Kahneman, A. Tversky (Eds.), Choices, Values and Frames, Cambridge University Press, New York, 2000.
[25] A.P. Kirman, Whom or what does the representative individual represent?, Journal of Economic Perspectives 6 (1992) 117–136
[26] A.P. Kirman, Economy as a Complex System, in: L.E. Blume (Ed.), The New Palgrave Dictionary of Economics, vol. 2, Palgrave Macmillian, 2008, pp. 723–729.
[27] F.W. Lawvere, R. Rosebrugh, Sets for Mathematics, Cambridge University Press, 2003.
[28] P. Martin-Löf, Intuitionistic type theory, Bibliopolis, 1984.
[29] A. Mas-Collel, The Theory of General Economic Equilibrium: A Differential Approach, Cambridge University Press, Cambridge, 1985.
[30] B. Meyer, Object-oriented Software Construction, second ed., Prentice-Hall Resource, Prentice-Hall, 2000.
[31] B.H. Mitra-Kahn, General Equilibrium Theory, its history and its relation (if any) to the Market Economy. Unpublished, 2005.
[32] B.H. Mitra-Kahn, Debunking the Myths of Computable General Equilibrium Models, Schwartz Center for Economic Policy Analysis working Paper 2008-1, 2008.
[33] C. Morgan, Programming from Specifications, second ed., Prentice-Hall International Series In Computer Science, Prentice-Hall, 1990.
[34] B. Nordström, K. Petersson, J. Smith, Programming in Martin-Löf's Type Theory, Oxford University Press, 1990.
[35] J. Gary Polhill, Dawn Parker, Daniel Brown, Volker Grimm, Using the ODD protocol for describing three agent-based social simulation models of land-use change, Journal of Artificial Societies and Social Simulation 11 (2) (2008) 3.
[36] S. Reiter, S. Maroulis, Stable processes of exchange, Journal of Mathematical Economics 44 (2008) 13981412.
[37] H. Scarf, Some examples of global instability of competitive equilibrium, International Economic Review 1 (1960) 157–172.
[38] I. Sommerville, Software Engineering, eighth ed., Addison-Wesley, 2006.

[39] H. Sonnenschein, Do Walras identity and continuity characterize the class of community excess demand functions?, Journal of Economic Theory 6 (1973) 345–354
[40] K. Taisei, Multiple equilibria and chaos in a discrete tâtonnement process. Technical report, International Christian University, MPRA Paper No. 24002, 2010.
[41] L. Tesfastion, K. Judd (Eds.), Handbook of Computational Economics II: Agent-Based Computational Economics, North-Holland, 2006.
[42] Hal R. Varian, Microeconomic analysis. Norton, third edition, 1992.
[43] L. Walras, Éléments d'économie politique pure, ou théorie de la richesse sociale (Elements of Pure Economics, or the theory of social wealth). Corbaz, Lausanne, 1874.