Proceedings of the 2015 Winter Simulation Conference
L. Yilmaz, W. K V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.

ML3: A LANGUAGE FOR COMPACT MODELING OF LINKED LIVES IN COMPUTATIONAL DEMOGRAPHY

Tom Warnke
Alexander Steiniger
Adelinde M. Uhrmacher

Institute of Computer Science
University of Rostock
Albert-Einstein-Str. 22
D-18059 Rostock, GERMANY

Anna Klabunde
Frans Willekens

Max Planck Institute for Demographic Research
Konrad-Zuse-Straße 1
D-18057 Rostock, GERMANY

ABSTRACT

Agent-based modeling and simulation is widely used in computational demography. Although existing agent-based approaches allow modeling linked lives in a rather flexible manner, the resulting models, due to typically being implemented in a general-purpose programming language, often lack the compactness required to easily access the model. With ML3 (Modeling Language for Linked Lives) we present a compact and expressive domain-specific modeling language for continuous-time agent-based models in computational demography. The language combines elements from guarded commands, process algebras, and rule-based approaches. We discuss and present the individual features of the language and illuminate its compactness by presenting the specification of an entire agent-based model from recent literature.

1 INTRODUCTION

During the last decade, demographic microsimulation moved steadily toward a more agent-based perspective. Agent-based models consider individuals, their decision processes, and interaction patterns in more detail than traditional microsimulation or object-oriented approaches typically do (Uhrmacher 1996). Even though individuals are assumed to make decisions in microsimulation models too, these are usually not "very explicit and detailed about the path [...] subjects follow to reach a decision" (Klevmarken 2008, p. 33). Widely cited examples of agent-based models in demography include Todd et al. (2005), Hills and Todd (2008), Diaz et al. (2011), Billari et al. (2007), and Fent et al. (2013). These models have in common that the interaction between individuals and "linked lives" — persons connected in couples or households — is of particular importance.

The increased complexity of agent-oriented models poses several challenges for modelers. For demographers who typically do not have a background in computer science, the development and implementation of an agent-based model can be difficult and error-prone. At the same time, the documentation of models becomes more important and less straightforward. Insufficient documentation hampers the re-implementation of models and ultimately scientific progress.

The validation of models is influenced by the shift toward agent-based modeling as well. As such models are structurally complex and rich in hypotheses, estimating parameters directly from data is often no longer possible. Instead, one has to resort to methods like simulation-based optimization that rely on experimenting with these models (Silverman et al. 2013).

Domain-specific languages (DSLs) can remedy many of these issues. For instance, languages for specifying experiments (Schützel et al. 2014, Ewald and Uhrmacher 2014) can help in tailoring experiments to the requirements of agent-based demographic models. Domain-specific modeling languages promise help in developing, documenting, revising, and reusing demographic agent-based models. In this paper, we propose a novel domain-specific modeling language for agent-based computational demography.

In order to be useful for demographers, a language has to allow for a stringent individual-centric approach, so that phenomena at the macro level, such as population change, population compositions and age distributions, can be derived from micro-behavior. However, there is not only the individual level to be considered, but also organizational units such as couples, households, and communities that have attributes of their own. Decision-making is not restricted to individuals, as it is often the case for agent-based models in social science, but also possible at the level of a couple regarding childbirth or divorce, or the household regarding migration. Thus, individuals must be linked to higher-order entities and each other in network-like structures. This has been done frequently in agent-based models in social science (Billari et al. 2007, Macy and Willer 2002). These links are dynamic and diverse, so that agents can e.g., begin and end friendships, get married, get divorced, or remarry.

Demographic models describe population change in terms of demographic events, such as death, marriage, divorce, childbirth and migration. Age and calendar time are common time scales. To estimate durations until events occur and duration between events, demographers use models that are closely related to models developed in statistics for the study of time-to-event data. This subject of study is commonly referred to as survival analysis or duration analysis (see e.g. Aalen et al. (2008)). Events and event sequences are viewed as outcomes of random processes in continuous time. It is desirable to tie simulation models to these probability models. Whereas discrete-event continuous-time microsimulation is fairly established, continuous-time agent-based models are not yet common. One of the reasons for this is that the vast majority of existing software packages for agent-based models do not allow for continuous-time but dictate a step-wise discrete time simulation approach.

2 RELATED WORK

For agent-based modeling and simulation a number of tools has been developed in the last decades. A recent survey by Kravari and Bassiliades (2015) reviews, classifies, and compares 24 platforms that support the development and execution of agent-based models. Some are general-purpose tools, others focus on a certain application domain. However, most tools do not introduce a new, specifically developed modeling language, but require that the modeler uses a general-purpose programming language, often Java or C/C++. The same applies if model and simulation algorithm are programmed from scratch, as has been done in the linked lives model by Noble et al. (2012). The interweaving of model and simulation code often hinders the understanding and reuse of a model additionally. Regardless of whether an agent-based modeling and simulation (ABM) tool is used, a modeler often needs to translate a conceptual model into an executable one based on a general-purpose programming language. However, as argued by Grimm et al. (2006), the implementation of models in a general-purpose programming language has several drawbacks: It is often not sufficiently understandable without extensive documentation, and length and complexity of the code makes including model code in publications unfeasible. This hampers the validation and reuse of models. To address this issue, ODD (Overview, Design concepts, and Details) has been introduced (Grimm et al. 2006). ODD is a protocol that aims to standardize the documentation of agent-based models. The description is structured into overview, design concepts, and details; the overall goal is to achieve a level of detail that allows recreating a model. However, an ODD model description is neither unambiguous nor executable. Consequently, the developers of ODD recommend publishing the original source code of the model together with the ODD description. Thus, ODD aims at enhancing the documentation and re-usability of agent-based models by bridging the gap between conceptual and executable model.

An alternative to close this gap are DSLs that provide a succinct and unambiguous description of a system. Some ABM systems support declarative languages, but are usually limited to a very domain-specific points of view. NetLogo (Tisue and Wilensky 2004) for example, being intended for educational use, includes a language that is easy to learn but offers little support for continuous-time models. Although it is argued that it is possible to implement various types of models in NetLogo, it is clearly designed for models with certain characteristics (e.g., grid space) (Railsback et al. 2006). SDML (Moss et al. 1998) and AORSL (Wagner 2010) are other declarative agent modeling languages for simulation. However, SDML does neither support continuous-time microsimulation nor allow organizational structures to have their own behavior; AORSL is based on XML and thus not as easily accessible as DSLs. Among the projects that have specifically developed a formal language for agent-based modeling for computational demography, MIMOSE (MIcro- and Multilevel MOdeling SoftwarE) (Möhring 1996) is one of the first. A simulation framework for the execution of models complements the modeling language, which aims at integrating features needed by social scientists while relieving modelers of programming. Many of the considerations in the development of the language are still relevant today. However, the execution of models is restricted to fixed stepwise approximations of differential equations.

A more recent project that targets the domain of computational demography with a tailored language is Modgen (Model generator) (Statistics Canada 2014). It allows specifying models with linked actors and executing them through microsimulation. The Modgen language builds upon C++ and therefore requires the modeler to have "some understanding" of this programming language. This has the same disadvantages as the already mentioned general-purpose approaches, and modelers with no background in computer science or programming will have difficulties in understanding or writing models. In addition, the resulting models lack the desired compactness. Another approach that brings together agent-based modeling and computational demography is JAMSIM (Mannion et al. 2012). However, it only supports discrete-time simulation and relies on the programming language Java to specify agent behavior.

Finally, there are some approaches that use domain-specific modeling languages from other domains for agent-based simulation in demography. In Steiniger et al. (2014), we applied three modeling formalisms, for example developed for computational systems biology, to recreate the linked lived model from Noble et al. (2012). The results of our work concluded that, whereas the model is expressible in all used languages, the readability and succinctness suffer due to properties of the model and the requirements of computational demography in general. Another application of a DSL from computational systems biology for agent-based computational social science is described in Kühn and Hillmann (2014). Here, the rule-based language BioNetGen is applied to implement a labor market model. Among the advantages of using BioNetGen are the unambiguity and succinctness of the model description and the clear execution semantics of the available algorithms. The identified disadvantages include the inability to make use of the model time and language features reflecting its cell biological background that did not fit into the social science domain. Thus, these findings confirm our conclusions in Steiniger et al. (2014).

Clearly, DSLs have a great potential for a succinct, unambiguous, and readable specification of models. In other domains, e.g., computational biology, DSLs are widely and successfully used (cf. Faeder (2011)). However, domain-specific declarative modeling languages for agent-based demographic models are scarce. It seems that language design in this domain was often willing to sacrifice simplicity for the sake of expressiveness, thus effectively limiting the circle of users to modelers with programming skills.

3 REQUIREMENTS AND DESIGN PRINCIPLES OF ML3

Modeling language design aims at maximizing expressiveness (referring to the requirements of a domain) while minimizing conceptual complexity, within the constraint of efficient execution. Thus, the yardstick of any modeling language are concrete requirements from the application domain. In this section, we analyze requirements for modeling languages in general as well as domain-specific modeling languages for computational demography and present how our proposed **M**odeling **L**anguage for **L**inked **L**ives (ML3) addresses these requirements. But first, a short overview of the language structure is given.

Language structure: The central model entity in ML3 are agents that are defined by agent types, related to each other by links and whose behavior is described by behavior rules (Figure 1). A behavior rule consists of a guard that determines if the rule is applicable for a concrete agent depending on its state, a rate expression that states how prone an agent is to show the behavior, and a list of behavior effects. Such stochastic guarded commands are argued to be a natural way to specify population models (Henzinger et al. 2011), but the advantages of this approach are also evident for agent-based modeling. Compared to pattern matching, the flexibility of guard statements allows an even more consequent adoption of a "don't-care-don't-write" attitude, thus facilitating succinct description of agents with many attributes.

```
⟨Model⟩      ::=  ⟨ImportStatement⟩* ⟨AgentType⟩* ⟨Link⟩* ⟨Function⟩* (⟨AgentName⟩ ⟨Rule⟩*)*
⟨AgentType⟩  ::=  ⟨AgentName⟩ '(' ⟨Attribute⟩* ')'
⟨Attribute⟩  ::=  ⟨AttributeName⟩ ':' ⟨AttributeType⟩ [':=' ⟨DefaultValue⟩]
⟨Link⟩       ::=  ⟨Role⟩ ':' ⟨AgentName⟩ '['⟨Card.⟩']' '↔' '['⟨Card.⟩']' ⟨Role⟩ ':' ⟨AgentName⟩
⟨Function⟩   ::=  ⟨AgentName⟩ '.' ⟨FunctionName⟩(⟨Argument⟩*) ':=' ⟨Expression⟩
⟨Rule⟩       ::=  '|' ⟨GuardExpression⟩* '@' ⟨RateExpression⟩ '→' ⟨EffectExpression⟩*
```

Figure 1: A snippet of the abstract syntax of ML3

The structure of ML3 is further illustrated in Section 4. Now, we analyze the requirements that guided the development of ML3 and describe central language design principles.

Expressiveness and conciseness: With expressiveness (or expressivity) we do not refer to whether something can in principle be expressed in a modeling language (e.g., we could program in a Turing machine, but this is hardly advisable); or in the words of Perlis (1982), p. 10: "Beware of the Turing tar-pit in which everything is possible but nothing of interest is easy." We refer to *practical expressivity* as defined by Farmer (2007): "the measure of how readily ideas can be expressed in the logic [language]" or, in other words, the feature of a language that allows us to express the significant concepts of a model, e.g., relationships or space, completely and with little clutter. With this in mind, a language can be considered expressive if it helps the modeler to express interesting aspects of a dynamic system in a concise and succinct manner. Therefore, (practical) expressiveness is closely related to the notion of conciseness or succinctness. Two different strategies can be distinguished in enhancing expressiveness in modeling languages: First, language constructs for specific dynamic phenomena can be introduced, e.g., the operations "split" and "join" in Beta Binders splitting and joining compartments (Priami and Quaglia 2005). Second, more general concepts can be employed, e.g., in Bigraphs by the composition of place graphs and link graphs (Milner 2001). Our experience is in developing domain-specific languages according to the latter strategy. This strategy is maybe most obvious in Attributed Pi (John et al. 2008), where processes of the stochastic pi-calculus are equipped with attributes and constraints that are defined based on the lambda calculus. Thus, concurrent processing, local names, and scoping of the pi-calculus are combined with higher order process passing by explicit substitution of the lambda calculus. Its successor language ML-Rules pursues the same strategy (Maus et al. 2011) and adds active processes in locations. It shows the value of this strategy, its flexibility (also arbitrary functions can be defined) (Warnke et al. 2015) and also the impact of adopting a domain-specific modeling metaphor.

ML3 supports the design of succinct and expressive models in several ways. Most importantly, the language is designed by a team that consists of both computer scientists and demographers. Thus, we are able to pick up domain concepts without losing sight of technical feasibility. Central domain abstractions such as agents, links, and behavior rules as well as functions that allow encapsulating recurring computations under a meaningful name are fixated in the structure of the model definition. Furthermore, ML3 includes some default elements that are frequently needed in demography. Agents (individuals and organizations) have predefined attributes such as *isAlive*, *birthDate*, *deathDate*, and *age* that are automatically updated during the simulation. Functions for creation and dissolution of agents are supplied as well. Data types such as boolean values, natural and real numbers, strings as well as dates are predefined. Finally, a clear separation of model definition and simulation algorithm allows hiding execution details. For example, our current prototypical simulation algorithm resolves stochasticity of the model through stochastic race.

Compositionality. "A compositional language allows the modular description of a system by combining submodels that describe parts of the system" (Henzinger et al. 2011), where submodels are represented by constructs of the modeling language. Thus, the *compositionality* of a language facilitates modeling complex systems, which often consist of interacting entities (system components). For instance, the modular, hierarchical structure of DEVS (Zeigler et al. 2000) explicitly supports the composition of models. Also, process algebras explicitly support composition by specific operators that belong to the language, e.g., processes can be defined by composing '|' two concurrent processes $p = p_1|p_2$. Rule based models (Faeder 2011) or Petri Net models (van der Aalst et al. 2013) may be joined to create larger ones. However, the identification of what is composed is lost, as species or places with the same name are merged. Thus, additional annotations are required. In ML3, models containing different agent types can be merged which leads to agents from the original models coexisting in the composed model. Interactions between agents can be implemented by adding corresponding rules, similar to adding interaction to processes in process algebras. Thus, ML3 supports compositionality.

Object of interest — the individual. In computational demography, the focus is on modeling individuals or organizations and their interaction. In an agent-based model, the entities are represented by *agents* and their interaction in the system is mapped to interaction between agents. A modeling language that focuses on individuals and organizations in such a way facilitates capturing decision processes of these entities. In ML3, we pursue a strict individual-based modeling approach. A model state consists solely of agents, i.e., their individual states and relations. Model dynamics are exclusively given by behavior rules of agent types; thus, changes in the model state can only be issued by acting individuals.

Attributes, constraints, and guarded commands. Agents have attributes that describe their general properties and that may change over time. These attributes determine the likelihood of events happening to an agent. For example, age and sex of an individual may determine the likelihood of finding a partner. To constrain possible interactions and events, in ML3 arbitrary functions can be used. We plan to ground these on the lambda calculus in a formal semantics for ML3. Thus, similarly as in Attributed Pi (John et al. 2008) or ML-Rules (Warnke et al. 2015), a powerful language to access attributes is supplied.

Individuals and organizations — tied together by links. In demography, individuals are not the only active entities. Higher organizational structures, such as families, communities, or associations, may exhibit a behavior of their own and be influenced by their members. Conversely, the behavior of individuals may be influenced by organizations and other individuals. Thus, the life courses of individuals can be *linked* with each other and with organizations over longer periods of time, e.g., as a result of an interaction between different individuals. These links have direct or indirect influence on the behavior of agents. Links between individuals can represent various kinds of relationships, such as kinship, marriage, friendship, or partnership, each of which may have a different degree of influence on the affected life courses. For instance, life changes of a spouse often have a greater influence than those of a colleague. Due to their importance, a strong concept of links is a requirement for a modeling language specific for agent-based computational demography. In ML3 links are first-class citizen and explicitly defined, clearly separated from non-link attributes of agents. Links are always bidirectional. On both ends, the number of agents participating in one link is defined (cardinality) and role names for the linked agents are given. These role names can then be used just like attributes of an agent to access agents on the other end of the link. As an agent's links can point to a set of agents, predefined functions to operate on agent sets are included.

Changing interactions and relations. As already stated in Uhrmacher (1996), variable structures are an intrinsic feature of social systems and have to be reflected in agent-oriented social models. Organisations, families, etc. can dissolve over time, same as links to other individuals and organizations. However, sometimes knowledge about even no longer existing structures is still important as it directs the behavior of individuals. In ML3 it is possible to keep links to "dead" agents, including dissolved organizations etc. While those agents will not show any behavior of their own anymore, they can still influence the behavior of living agents.

Age dependent behavior. Age plays a special role in population dynamics in general, e.g., by different juvenile and adult mortality rates, and in demography in particular. For instance, age-specific rates can often be found in agent-based demographic models such as Billari et al. (2007) and Lawson (2009). Age

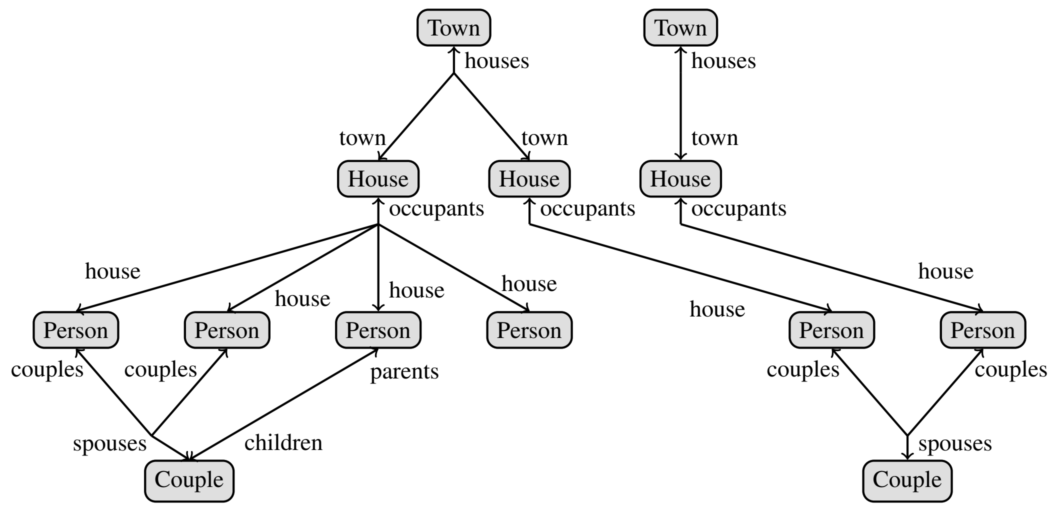

Figure 2: Snippet of a state of our linked lives model, showing the relations between different agent types.

norms can also guard certain actions such as marrying or adopting. ML3 supports the modeler by supplying a predefined age attribute for all agents and automatically advancing it as the simulation progresses.

4 WORKING EXAMPLE

In this section we will illustrate how the linked lives model (Noble et al. 2012) can be expressed in ML3. As an agent-based model, it includes demographic processes like birth, death, and internal migration but also inter-agent processes such as forming and dissolving partnerships and families. The agents live in houses that are located in towns, whereas the towns are placed on a grid. All agents are born healthy; as they get older, the risk of their health level degrading rises. The model is used to predict the population structure of the UK and the development of need and supply of social care. It is computed in discrete steps of one year, with stochastic events for each agent at the end of each year. The original Python implementation of the model is available online. To showcase our proposed language ML3, we used it to express this model. We discarded code for controlling the execution of the model, the observation (graphical output), and analysis of trajectories (statistical evaluation) as this is outside of the scope of a modeling language, and refers to the experiments done with the model. Furthermore, we also decided to move the description of the initial population and other model parameters out of the specification of the model dynamics to extra files. In the end, due to the conciseness of ML3, we were able to save a large amount of space compared to the corresponding Python code.

We made some adjustments concerning the structure of the model, too. First, we introduced a new type of agent representing a couple to model behavior of married persons. Divorce or having a child for example can thus be expressed from the point of view of the couple and in dependence of its attributes, as opposed to the original model where these behaviors are expressed from the perspective of an individual. The dynamics still follow the original model, though; e.g., divorce is initiated by males exclusively and only depends on their attributes. Second, we replaced the probabilities of the stepwise model with hazard rates of events in a continuous-time model. For instance, interpreting the stepwise development as Bernoulli experiment, i.e., geometrically distributed with parameter p, the rate $\lambda = -\log(1-p)$ of a corresponding exponential distribution can be obtained from the cdfs of both distributions. This facilitates a more precise timing of events. For example, in the continuous-time model it is possible for a person to marry and later die in the same year. As death events occur before other events at the end of each year in the discrete-time model, this is not possible there. A snippet of a model state is shown in Figure 2.

We will now walk through the complete model definition in ML3.

```
 1 import("parameters.dat", "initPop.dat");
```

In line 1 model parameters are imported from files, which makes them available as global constants. Model parameters can, for example, include the initial population, transition matrices, or scaling factors.

```
 3 Person(sex:{"m","f"},
 4   status:{"child", "adult at home", "independent adult", "retired"} := "child",
 5   careNeedLevel:nat := 0,
 6   sec:nat); // socio-economic category
 7
 8 Couple();
 9
10 House(sec:nat); // socio-economic category
11
12 Town(x:nat, y:nat);
```

Lines 3–12 declare four different types of agents: person, couple, house and town. Each agent has a fixed number of named attributes that describe its state, e.g., a person has four attributes. For each attribute a value range must be specified by either using a predefined type (e.g., `nat` for natural numbers), or by enumerating all possible values (e.g., `status`). Default values can optionally be given for each attribute.

```
14 parents:Couple[1] <-> [0-]Person:children;
15 couples:Couple[0-] <-> [2]Person:spouses;
16 house:House[1] <-> [0-]Person:occupants;
17 town:Town[1] <-> [1-]House:houses;
```

The link declarations in lines 14–17 describe the relations between agent types. Links are always bidirectional, multiplicity and role names must be given for both ends. An agent's links can dynamically change. However, at all times, links must be connected to a correct number of agents of the correct type. If a link is deleted or created from one side, the other side is changed accordingly. In line 14 the relation between children and parents is described. Each person has a link to exactly one couple, his parents. Each couple can have an arbitrary number of children, including zero. If a person is linked to a couple as its child, this couple is also linked to the person as his parents.

```
19 Person.single() := ego.couples.filter(alter.isAlive()).size() = 0;
20
21 Person.activeCouple() := ego.couples.filter(alter.isAlive()).only();
22
23 Person.allChildren() := (ego.couples.sum(alter.children)).filter(alter.isAlive(),
       alter.status = "child");
24 Person.withAllChildrenHere() := ego + ego.allChildren().filter(alter.house =
       ego.house);
25
26 Person.deathRate() := baseDieRate +
27   (ego.age < 1 ? babyDieRate : 0) +
28   (ego.age / ageScaling[ego.sex]) * ageDieRate[ego.sex];
29
30 Person.marriageRate(?woman : Person) :=
31   basicFemaleMarriageRate * femaleMarriageModifier[?woman.age] +
32   basicMaleMarriageRate * maleMarriageModifier[ego.age];
33
34 Person.moveToChildRate(?child : Person) :=
35   (now < thePresent ? agingParentsMoveInWithKids : variableMoveBack) /
36   (dist(ego.house.town, ?child.house.town) + 1);
37
38 Person.findNewHouseFor(?persons : [Person], ?preference : string) :=
39   if (?hereHouse != none && ?preference = "here") then ?hereHouse.occupants +=
         ?persons
```

```
40   else if (?nearHouse != none && (?preference = "near" || ?preference = "here")) then
         ?nearHouse.occupants += ?persons
41   else if (?farHouse != none && (?preference = "far" || ?preference = "near" ||
         ?preference = "here")) then ?farHouse.occupants += ?persons
42     where ?hereHouse := ego.house.town.houses.filter(alter.pop() = 0, alter.sec =
           ego.sec).random(),
43       ?nearHouse := House.all.filter(alter.town.neighbour(ego.house.town),
             alter.pop() = 0, alter.sec = ego.sec).random(),
44       ?farHouse  := House.all.filter(alter.pop() = 0, alter.sec = ego.sec).random();
```

Functions can be defined for agent types. They can then be invoked for a concrete agent and access his attributes, links, and functions by using the keyword `ego` and the point operator. Accesses to attributes, links, and functions can also be chained by the point operator. Further, predefined set functions such as `filter` or `size` can be used. The keyword `ego` is a well known metaphor in computational demography for an acting agent, as is `alter` for an interaction partner (Snijders et al. 2010). Lines 19 to 44 show functions for persons. In line 19 for example, the function `single()` is defined to evaluate to true if a person has no link to an active couple. Functions can also be provided with arguments. In lines 38 to 44, a function `findNewHouseFor` for the agent type `Person` is defined that takes into account a group of persons and a string. Depending on the value of the string, it moves the given persons to an empty house that is in the same town, a neighbouring town, or any town according to the place of residence of the person the function is invoked on. The new house must have the same socio-economic category as this person. Also, this function shows how an access chain can be started from an agent type followed by the keyword `all` to access the set of all agents of that type currently alive in the model. In line 44 for example, the expression `House.all.filter(alter.pop() = 0, alter.size = ego.sec).random()` will evaluate to a random house that is empty and has the same socio-economic category as the person looking for a house. If such a house does not exist, it will be `none`. The function `findNewHouseFor` illustrates the complexity of decision processes in demographic models and the necessity for a suitable support in the modeling language.

```
46 Couple.splitRate() := divorceModifier[ego.husband().age] *
47   (now < thePresent ? basicDivorceRate : variableDivorce);
48
49 Couple.bothDead() := ego.spouses.filter(alter.isAlive()).size() = 0;
50
51 Couple.wife() := ego.spouses.filter(alter.sex = "f").only();
52 Couple.husband() := ego.spouses.filter(alter.sex = "m").only();
53
54 House.pop() := ego.occupants.size();
55
56 Town.distX(?other : Town) := abs(ego.x - ?other.x);
57 Town.distY(?other : Town) := abs(ego.y - ?other.y);
58 Town.dist(?other : Town)  := ego.distX(?other) + ego.distY(?other);
59 Town.neighbour(?other : Town) := ego.distX(?other) <= 1 && ego.distY(?other) <= 1;
```

Lines 46–59 show functions of other agent types. For example, functions `wife()` and `husband()` of a couple filter the linked spouses by sex and return the first (and only) person in the resulting set. In line 54 the function `pop()` for the type `House` is defined to return the number of occupants of a house.

```
61 Person
62
63   // death
64   | @ ego.deathRate() -> ego.death(), ego.activeCouple().death();
65
66   // care transition
67   | ego.careNeedLevel < numCareLevels - 1
68   @ baseCareRate + (ego.age / ageCareScaling[ego.sex]) * ageCareRate[ego.sex]
```

```
69   -> ego.careNeedLevel := min(ego.CareNeedLevel + careTransition[rnd()],
         numCareLevels);
70
71   // age transition
72   | ego.status = "child" @ ego.age(ageOfAdulthood) -> ego.status := "adult at home";
73   | ego.status = "adult at home", ego.parents.bothDead() @ instantly -> ego.status :=
         "independent adult";
74   | ego.status != "retired" @ ego.age(ageOfRetirement) -> ego.status := "retired";
75
76   // getting adopted
77   | ego.status = "child", ego.parents.bothDead()
78   @ instantly
79   -> ego.parents := Couple.all.random();
80
81   // marriage
82   | ego.status != "child", ego.sex = "m", ego.single()
83   @ ego.marriageRate(?candidate)
84   for each ?candidate : Person.all.filter(alter.isAlive(), alter.sex = "f",
         alter.single(), alter.parents != ego.parents, ego.age - alter.age < 20, ego.age
         - alter.age > -5)
85   -> new Couple(spouses := [ego, ?candidate]);
86
87   // move
88   | ego.single(), ego.status = "adult at home"
89   @ basicAdultMoveOutRate * adultMoveOutModifier[ego.age]
90   -> ego.findNewHouseFor(ego.withAllChildrenHere(), ["here", "near"].random());
91   | ego.single(), ego.status = "independent adult"
92   @ basicSingleMoveRate * singleMoveModifier[ego.age]
93   -> ego.findNewHouseFor(ego.withAllChildrenHere(), ["here", "near"].random());
94   | ego.status = "retired", ego.house.occupants.size() = 1
95   @ ego.moveToChildRate(?child)
96   for each ?child := ego.allChildren()
97   -> ego.house := ?child.house;
```

Finally, in lines 61 to 97 behavior rules for persons are defined. Behavior rules always consist of three parts: Conditions that guard the applicability (started by `|`), a hazard rate expression (started by `@`) and a number of effects (started by `->`). The first rule (line 64) describes the risk of death for a person. It has no guard conditions, i.e., a person is always exposed to the risk of death. The hazard rate is determined by the function `deathRate()` that was defined earlier. If this rule is applied, two effects follow: The agent dies, and his active couples dies. In this model, there will be at most one active couple per person, and couples are only active if both spouses are alive. Dead couples however, i.e., couples dissolved through divorce or death of a spouse, can still be parents of persons. Also, agents keep links to inactive couples, for example to maintain (indirect) links to children from former marriages.

The rule for care transition (lines 67–69) is guarded by the condition that the concerned agent has not already reached the highest care level. The rate is then dependent on the age and some sex-specific constants. As an effect, the level of care need is raised, but not higher than the maximum care need level.

The rules for age transition in lines 72–74 showcase another feature that is central for demographic modeling. Here, the rate expressions contain the keyword `age()`, meaning that the rule will be applied to all agents that satisfy its conditions upon reaching the given age. The keyword `instantly` is a generalization of this, meaning that a rule will be applied as soon as an agent satisfies its conditions.

In lines 82–85 a rule for marriage is defined. Here, each agent in the set of potential partners for a person is assigned to the variable `?candidate`. For each potential partner, a hazard rate for marriage is determined, i.e., the rule is instantiated multiple times. The application of the rule results in the creation of a new couple, with its spouses link set to the initially concerned agent and the chosen candidate.

```
99   Couple
```

```
100
101   // birth
102   | ?wife.age > minPregnancyAge, ?wife.age < maxPregnancyAge
103     @ now < transitionYear ? growingPopBirthRate : steadyPopBirthRate
104     -> new Person(sex := ["m", "f"].random(), sec := ?wife.sec, parents := ego, house
           := ?wife.house)
105     where ?wife := ego.wife();
106
107   // divorce
108   | @ ego.splitRate()
109   -> ego.death(), ego.husband().findNewHouseFor([ego.husband()]);
110
111   // whole family moving
112   | ego.wife().house = ego.husband().house
113   @ basicFamilyMoveRate * familyMoveRateModifier[ego.age]
114   -> ego.husband().findNewHouseFor(ego.husband().withAllChildrenHere() +
          ego.wife().withAllChildrenHere(), ["here", "near", "far"].random());
115
116   // move in together
117   | ego.wife().house != ego.husband().house
118   @ moveTogetherRate
119   -> if rnd() < coupleMovesToExistingHousehold then
120         if ego.wife().house.pop() < ego.husband().house.pop()
121           then ego.wife().house.occupants += ?husbandPersons
122         else ego.husband().house.occupants += ?wifePersons
123       else ego.husband().findNewHouseFor(?husbandPersons + ?wifePersons, ["here",
            "near"].random())
124   where ?wifePersons := ego.wife().withAllChildrenHere(), ?husbandPersons :=
          ego.husband().withAllChildrenHere();
```

The final part of the model comprises rules for couples (lines 99–124). In line 102, the birth of a child is modeled as the creation of a new person agent, including the initialization of its attributes and links. The hazard rate of a birth depends on the current date in the model, accessed with the keyword `now`.

5 CONCLUSION

We presented the concrete syntax of the modeling language ML3 aimed at describing continuous-time stochastic agent-based models in demography. Rather than presenting the complete syntax and semantics formally, we decided on an informal case-based approach to illustrate the practical expressivity of ML3. Applying ML3 to a concrete linked lives model allowed us to test certain design decisions, e.g., treating links as first class citizens of our language and thus making agents, links and behavior rules the focus of ML3. We demonstrated that these decisions resulted in a language that facilitates expressive model descriptions. At some points our model suffered from using an existing model rather than the original problem as a starting point, e.g., whereas in our approach a couple could easily induce a divorce, we reserved the initiation of divorces to the males of a relationship as done in the original model. Our next steps will be to define the formal syntax and semantics, and to complete the execution algorithm for ML3. This will serve as basis to apply ML3 in its first demographic simulation study.

ACKNOWLEDGEMENTS

This research is partly supported by the German Research Foundation (DFG) via research grant UH-66/15-1.

REFERENCES

Aalen, O. O., Ø. Borgan, and H. K. Gjessing. 2008. *Survival and Event History Analysis: A Process Point of View*. Statistics for Biology and Health. New York: Springer New York.

Billari, F., A. Prskawetz, B. Aparicio Diaz, and T. Fent. 2007. "The "Wedding-Ring": An Agent-Based Marriage Model Based on Social Interaction". *Demographic Research* 17 (3): 59–82.

Diaz, B. A., T. Fent, A. Prskawetz, and L. Bernardi. 2011. "Transition to Parenthood: The Role of Social Interaction and Endogenous Networks". *Demography* 48 (2): 559–579.

Ewald, R., and A. M. Uhrmacher. 2014. "SESSL: A Domain-specific Language for Simulation Experiments". *ACM Transactions on Modeling and Computer Simulation* 24 (2): 11:1–11:25.

Faeder, J. R. 2011. "Toward a Comprehensive Language for Biological Systems". *BMC Biology* 9 (1): 1–5.

Farmer, W. M. 2007. "Chiron: A Multi-Paradigm Logic". *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec, Studies in Logic, Grammar and Rhetoric* 10 (23): 1–19.

Fent, T., B. Aparicio Diaz, and A. Prskawetz. 2013. "Family Policies in the Context of Low Fertility and Social Structure". *Demographic Research* 29 (37): 963–998.

Grimm, V., 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. Jørgensen, W. M. Mooij, B. Müller, G. Pe'er, C. Piou, S. F. Railsback, A. M. Robbins, M. M. Robbins, E. Rossmanith, N. Rüger, E. Strand, S. Souissi, R. A. Stillman, R. Vabø, U. Visser, and D. L. DeAngelis. 2006. "A Standard Protocol for Describing Individual-Based and Agent-Based Models". *Ecological Modelling* 198 (1-2): 115–126.

Henzinger, T. A., B. Jobstmann, and V. Wolf. 2011. "Formalisms for Specifying Markovian Population Models". *International Journal of Foundations of Computer Science* 22 (4): 823–841.

Hills, T., and P. Todd. 2008. "Population Heterogeneity and Individual Differences in an Assortative Agent-Based Marriage and Divorce Model (MADAM) Using Search with Relaxing Expectations". *Journal of Artificial Societies and Social Simulation* 11 (4): 5.

John, M., C. Lhoussaine, J. Niehren, and A. M. Uhrmacher. 2008. "The Attributed Pi Calculus". In *Proceedings of the CMSB 2008*, 83–102. Berlin, Heidelberg: Springer-Verlag.

Klevmarken, A. 2008. "Dynamic Microsimulation for Policy Analysis: Problems and Solutions". In *Simulating an Ageing Population: A Microsimulation Approach Applied to Sweden*, edited by A. Klevmarken and B. Lindgren, Volume 285 of *Contributions to Economic Analysis*, 31–53. Elsevier.

Kravari, K., and N. Bassiliades. 2015. "A Survey of Agent Platforms". *Journal of Artificial Societies and Social Simulation* 18 (1): 11.

Kühn, C., and K. Hillmann. 2014. "Rule-Based Modeling of Labor Market Dynamics: An Introduction". *Journal of Economic Interaction and Coordination*.

Lawson, T. 2009. "A Demographic Microsimulation Model Using NetLogo". *Centre for Research in Economic Sociology and Innovation Working Paper Series* (2009-02): 21.

Macy, M. W., and R. Willer. 2002. "From Factors to Actors: Computational Sociology and Agent-Based Modeling". *Annual Review of Sociology* 28:143–166.

Mannion, O., R. Lay-Yee, W. Wrapson, P. Davis, and J. Pearson. 2012. "JAMSIM: A Microsimulation Modelling Policy Tool". *Journal of Artificial Societies and Social Simulation* 15 (1): 8.

Maus, C., S. Rybacki, and A. M. Uhrmacher. 2011. "Rule-Based Multi-Level Modeling of Cell Biological Systems". *BMC Systems Biology* 5 (1): 166.

Milner, R. 2001. "Bigraphical Reactive Systems". In *Proceedings of the CONCUR 2001*, 16–35. London, UK: Springer-Verlag.

Moss, S., H. Gaylard, S. Wallis, and B. Edmonds. 1998. "SDML: A Multi-Agent Language for Organizational Modelling". *Computational & Mathematical Organization Theory* 4 (1): 43–69.

Möhring, M. 1996. "Social Science Multilevel Simulation with MIMOSE". In *Social Science Microsimulation*, edited by K. G. Troitzsch, U. Mueller, G. N. Gilbert, and J. Doran, 123–137. London, UK: Springer.

Noble, J., E. Silverman, J. Bijak, S. Rossiter, M. Evandrou, S. Bullock, A. Vlachantoni, and J. Falkingham. 2012. "Linked Lives: The Utility of an Agent-based Approach to Modeling Partnership and Household Formation in the Context of Social Care". In *Proceedings of the WSC 2012*, 93:1–93:12.

Perlis, A. J. 1982. "Epigrams on Programming". *ACM SIGPLAN Notices* 17 (9): 7–13.

Priami, C., and P. Quaglia. 2005. “Beta Binders for Biological Interactions”. In *Proceedings of the CMSB 2004*, 20–33. Berlin, Heidelberg: Springer-Verlag.

Railsback, S. F., S. L. Lytinen, and S. K. Jackson. 2006. “Agent-Based Simulation Platforms: Review and Development Recommendations”. *Simulation* 82 (9): 609–623.

Schützel, J., D. Peng, A. M. Uhrmacher, and L. F. Perrone. 2014. “Perspectives on Languages for Specifying Simulation Experiments”. In *Proceedings of the WSC 2014*, 2836–2847.

Silverman, E., J. Bijak, J. Hilton, V. D. Cao, and J. Noble. 2013. “When Demography Met Social Simulation: A Tale of Two Modelling Approaches”. *Journal of Artificial Societies and Social Simulation* 16 (4): 9.

Snijders, T. A. B., G. G. van de Bunt, and C. E. G. Steglich. 2010. “Introduction to Stochastic Actor-Based Models for Network Dynamics”. *Social Networks* 32 (1): 44–60.

Statistics Canada 2014. “Modgen Version 10.1.0 Developer’s Guide”. http://www.statcan.gc.ca/sites/default/files/dev-guide-eng.pdf.

Steiniger, A., A. M. Uhrmacher, S. Zinn, J. Gampe, and F. Willekens. 2014. “The Role of Languages for Modeling and Simulating Continuous-Time Multi-Level Models in Demography”. In *Proceedings of the WSC 2014*, 2978–2989.

Tisue, S., and U. Wilensky. 2004. “NetLogo: Design and Implementation of a Multi-Agent Modeling Environment”. In *Proceedings of the Agent ’04*, 7–9.

Todd, P. M., F. C. Billari, and J. Simao. 2005. “Aggregate Age-at-Marriage Patterns From Individual Mate-Search Heuristics”. *Demography* 42 (3): 559–574.

Uhrmacher, A. 1996. “Object-Oriented and Agent-Oriented Simulation: Implications for Social Science Application”. In *Social Science Microsimulation*, edited by K. G. Troitzsch, U. Mueller, G. N. Gilbert, and J. Doran, 432–447. London, UK: Springer.

van der Aalst, W. M. P., C. Stahl, and M. Westergaard. 2013. “Strategies for Modeling Complex Processes Using Colored Petri Nets”. *Transactions on Petri Nets and Other Models of Concurrency* 7:6–55.

Wagner, G. 2010. “Model-driven Engineering of Second-Life-style Simulations”. In *Proceedings of the WSC 2010*, 791–798.

Warnke, T., T. Helms, and A. M. Uhrmacher. 2015. “Syntax and Semantics of a Multi-Level Modeling Language”. In *Proceedings of the ACM SIGSIM PADS 2015*, 133–144. New York: ACM.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press.

AUTHOR BIOGRAPHIES

TOM WARNKE is a Ph.D. student in the modeling and simulation group at the university of Rostock. His email address is tom.warnke@uni-rostock.de.

ANNA KLABUNDE is a research scientist at the Max Planck Institute for Demographic Research in Rostock in the research group ‘International Migration’. Her email address is klabunde@demogr.mpg.de.

ALEXANDER STEINIGER is a Ph.D. student in the modeling and simulation group at the University of Rostock. His email address is alexander.steiniger2@uni-rostock.

FRANS WILLEKENS is former director and Honorary Fellow of the Netherlands Interdisciplinary Demographic Institute (NIDI), The Hague and Emeritus Professor of the University of Groningen. He is also Chief Research Coordinator at the Max Planck Institute for Demographic Research. His email address is willekens@demogr.mpg.de.

ADELINDE M. UHRMACHER is professor at the Institute of Computer Science, University of Rostock and head of the modeling and simulation group. Her email address is adelinde.uhrmacher@uni-rostock.de.