# How to do social simulation in logic: modelling the segregation game in a dynamic logic of assignments

Benoit Gaudou, Andreas Herzig, Emiliano Lorini, and Christophe Sibertin-Blanc

University of Toulouse, France
UMR 5505, Institut de Recherche en Informatique de Toulouse (IRIT), CNRS, France
IRIT, Université Paul Sabatier, 118 Route de Narbonne, F-31062 TOULOUSE CEDEX 9
{benoit.gaudou,christophe.sibertin-blanc}@univ-tlse1.fr
{emiliano.lorini,andreas.herzig}@irit.fr

**Abstract.** The aim of this paper is to show how to do social simulation in logic. In order to meet this objective we present a dynamic logic with assignments, tests, sequential and nondeterministic composition, and bounded and non-bounded iteration. We show that our logic allows to represent and reason about a paradigmatic example of social simulation: Schelling's segregation game. We also build a bridge between social simulation and planning. In particular, we show that the problem of checking whether a given property $P$ (such as segregation) will emerge after $n$ simulation moves is nothing but the planning problem with horizon $n$, which is widely studied in AI: the problem of verifying whether there exists a plan of length at most $n$ ensuring that a given goal will be achieved.

## 1 Introduction

In a recent debate Edmonds [9] attacked what he saw as "empty formal logic papers without any results" that are proposed in the field of multi-agent systems (MASs). He opposed them to papers describing social simulations: according to Edmonds the latter present many experimental results which are useful to better understand social phenomena, while the former kind of papers only aim at studying some relevant concepts and their mathematical properties (axiomatization, decidability, etc.), while not adding anything new to our understanding of social phenomena. In response to Edmonds's attack, some researchers defended the use of logic in MAS in general, and in particular in *agent-based social simulation* (ABSS) [10, 5, 7]. For example, in [5] it is argued that logic is relevant for MAS because it can be used to construct a much needed formal social theory. In [10] it is argued that logic can be useful in ABSS because a logical analysis based on (a) a philosophical or sociological theory, (b) observations and data about a particular social phenomenon, and (c) intuitions — or a blend of them — can be considered to provide the requirements and the specification for an ABSS system and more generally a MAS. Moreover, a logical system might help to check the validity of the ABSS model and to adjust it by way of having a clear understanding of the formal model underpinning it. All these researchers consider logic and ABSS not only as compatible, but also as complementary methodologies.

The idea we defend in this paper is much more radical than those of the above advocates of logic-based approaches. Our aim is to show that ABSS can be directly done in logic and that a logical specification of a given social phenomenon can be conceived as an ABSS model of this phenomenon. We believe that the use of adequate theorem provers will allow to obtain results that are beyond the possibilities of existing simulators. As a first step towards our aim we present in this paper a simple logic called DL-PA (Dynamic Logic of Propositional Assignments). DL-PA is an extension of propositional logic with dynamic operators. These operators allow to reason about *assignments* $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ changing the truth value of a propositional variable $p$ to 'true' or 'false' and about *tests* $\Phi?$ of the truth of a Boolean formula $\Phi$. More generally, DL-PA allows to reason about those facts that will be true after complex events $\pi$ that are built from assignments and tests by means of the operators of sequential composition ($\pi_1;\pi_2$), nondeterministic composition ($\pi_1 \cup \pi_2$), bounded iteration ($\pi^{\leq n}$), and unbounded iteration ($\pi^{<\infty}$).

In order to illustrate the power of our logic we show that a paradigmatic ABSS model can be represented in our logic: Schelling's segregation game [17]. The problem of checking whether (under some initial conditions) a given property $P$ such as segregation will *possibly emerge* after $n$ simulation moves is reduced to the problem of checking in our logic whether the initial conditions imply that formula $\varphi$ encoding property $P$ will be true at the end of at least one sequence of events $\pi$ of length at most $n$. Similarly, the problem of checking whether $P$ will *necessarily emerge* after $n$ simulation moves is reduced to the problem of checking whether the initial conditions imply that $\varphi$ will be true at the end of every sequence of events $\pi$ of length $n$. Actually the latter is nothing but the planning problem with horizon $n$, which is widely studied in AI and is for example at the base of the state of the art planner SatPlan [15]: the problem of verifying whether there exists a plan of length at most $n$ ensuring that a given goal $\varphi$ will be achieved. In the general case this problem is known to be in PSPACE, *i.e.* decidable in polynomial space [3]. We show that our logic fits these boundaries. In the past such PSPACE hard decision problems were considered to be out of reach of automated theorem provers. However, in the last 20 years huge progress was made on that kind of problems: state-of-the-art theorem provers for PSPACE complete problems were shown to be of practical interest in particular in semantic web applications even for realistic problem instances with thousands of clauses [14].

One might wish to go beyond the simple existential and universal quantifications that we mentioned above. This can be achieved by means of modal operators with counting (stemming from graded modal logics [11, 21] and description logics [1]). We briefly discuss this extension of DL-PA and show that complexity of the star-free fragment remains in PSPACE.

Simulation and logic use quite different terminologies. Table 1 summarizes the correspondences between the concepts used in simulation and those used in logic. Note that the term 'model' occurs in both terminologies, but has different meanings: in simulation a model stands for a formal or conceptual model of a particular application one wishes to investigate, such as Schelling's segregation game in our case; in DL-PA, a model of

a formula is a valuation where that formula is true.[1] A further difference between both fields is that logical formulas allow to talk about the whole search space (e.g. about what may necessarily emerges in all paths through the space of possible paths), while simulation is only about a single path in the search space.

| simulation | dynamic logic |
|---|---|
| model | logical language + domain laws |
| state | state, valuation (in DL-PA also called a model) |
| individual action | atomic event, atomic program |
| simulation step | complex event, complex program |
| property | logical formula |
| state has a property | valuation is a model of a formula |
| model has a property | domain laws imply formula |
| simulation engine | theorem prover |

**Table 1.** Terminologies in simulation and in logic

The rest of the paper is organized as follows. In Section 2 we describe the segregation game. In Section 3 we define our basic logic, and in Section 4 we show how it allows to reason about the segregation game. Finally we sketch some extensions (Section 5) and conclude (Section 6).[2]

## 2 The segregation game

In this section we give an informal description of the segregation game. A formal description is given in Section 4.

### 2.1 The original model

Thomas C. Schelling in [17] studied the phenomenon of segregation and in particular the conditions of its occurrence due to "discriminatory individual choices" in groups with recognizable distinctions such as sex, age, colour, etc. The best-known example is

[1] The identification of a model with a valuation is therefore just as in propositional logic, and is proper to our logic DL-PA (and more generally to logics of propositional control). The kind of models that are used in standard dynamic and temporal logics are more complex transition systems having a set of possible worlds, a transition relation between possible worlds, and a valuation for each possible worlds.

[2] In the MABS pre-proceedings version of this article we gave a more general PSPACE complexity result for the whole DL-PA, but the proof turned out to be incorrect. We conjecture that the problem of DL-PA model checking (allowing for formulas with the $\pi^{<\infty}$ operator) is in fact EXPTIME hard. Specific solvers exist for problems in this complexity class, but they are much less efficient than those for PSPACE problems. We thus chose to consider only the star-free fragment of DL-PA in this paper.

the formation of color-dependent residential areas, under the influence of the individual preference of being surrounded by at most a threshold number of neighbours with different colour: above the threshold inhabitants are unhappy and will move to another location.

One of the main results of Schelling's work is to show that the segregation phenomenon emerges even with a quite high tolerance threshold. For example, even if each inhabitant accepts that the majority of the neighbours surrounding him has a colour different from his, there will nevertheless be a tendency to form groups of inhabitants with the same colour.

### 2.2 The implemented model

The segregation model has been implemented in many languages and formalisms, in particular in almost all agent-based simulation platforms. Good examples are NetLogo [24, 23] and GAMA [19]. Two mains implementations have been proposed for this model: cellular automaton models (where the cells are the active entities) and agent-based models (where inhabitants are represented as agents and can move from one cell to another). We represent here the Segregation model as a cellular automaton and will discuss in the conclusion how to model the agent-based version.

The global environment of the simulation is taken as a chessboard-like grid $N \times N$. Each of its cells is represented by a couple of integers $(k, l) \in [1..N] \times [1..N]$. A cell is either red (inhabited by a red agent), or blue (inhabited by a blue agent), or has no colour (uninhabited). When an unhappy inhabitant moves from one place to another free one then the latter takes the colour of the former and the former becomes colourless.

The two main parameters of the simulation are:

- the number of inhabitants $|\mathbb{A}|$ and
- the tolerance threshold: the number of different inhabitants from which on an inhabitant is unhappy, which is supposed to be the same for every inhabitant.

(Alternatively the parameters may be the density of inhabitants and the percentage of different inhabitants. We also note that most simulation models rather use the inverse of the tolerance threshold, called the similarity threshold.)

There is a scheduler which generates at each simulation step a random ordering of the set of cells and then activates the cells according to that ordering during the step. Upon activation a cell checks its happiness: a cell is happy iff the percentage of neighbour cells having a different colour is below its tolerance threshold. If the cell is unhappy then its inhabitant will move to another free cell on the grid.

The simulation stops when a stable state is reached. This is the case when every coloured cell (*i.e.* every inhabitant) is happy.

A simulation may have three different behaviours: (1) the simulation loops because the system does not reach a stable state where every cell is happy (typically when the density of inhabitants on the grid is high and the tolerance threshold is low, which means inhabitants are very intolerant); (2) the simulation stops but one cannot observe any kind of segregation (this is typically the case when the similarity threshold is very low, *i.e.* tolerance is high and/or density is very low); or (3) clusters of cells with the same colour emerge.

## 3 Dynamic logic with assignments

This section introduces the syntax and the semantics of the logic DL-PA. It is basically an instantiation of propositional dynamic logic PDL [13] with concrete programs $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ assigning propositional variables to either true or false.

### 3.1 Language

We suppose given a countable set of propositional variables $\mathbb{P}$ with typical elements $p, q, \ldots$ Remember that the set of Boolean formulas of classical propositional logic can be built from $\mathbb{P}$ by means of the Boolean operators of negation and disjunction (the other connectives being defined by means of abbreviations).

For Schelling's game we will use evocative variables such as $\mathsf{R}(k,l)$ and $\mathsf{B}(k,l)$. Boolean formulas allow to express things such as $\bigwedge_{(k,l)} \neg(\mathsf{R}(k,l) \wedge \mathsf{B}(k,l))$, representing the fact that a cell cannot be both red and blue.

The set of *events* or *programs* $\Pi$ is defined by the following BNF:

$$\pi ::= p{\leftarrow}\top \mid p{\leftarrow}\bot \mid \Phi? \mid \pi;\pi \mid \pi{\cup}\pi \mid \pi^{=n} \mid \pi^{\leq n} \mid \pi^{<\infty}$$

where $p$ ranges over $\mathbb{P}$, $\Phi$ ranges over the set of Boolean formulas, and $n$ ranges over the set of natural numbers $\mathbb{N}$.

The events $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ are assignments modifying the truth value of the propositional variable $p$: the event $p{\leftarrow}\top$ sets $p$ to true, and the event $p{\leftarrow}\bot$ sets $p$ to false. $\Phi?$ is the test of $\Phi$, which fails if $\Phi$ is false. $\pi_1;\pi_2$ denotes a sequence of events. $\pi^{=n}$ denotes iteration of $\pi$ exactly $n$ times, $\pi^{\leq n}$ denotes iteration of $\pi$ up to $n$ times, and $\pi^{<\infty}$ denotes arbitrary iteration of $\pi$. Assignments and tests are atomic events, while the other events are called complex. An example of a complex event is a blue inhabitant's move from location $(k,l)$ to location $(k',l')$, written as the sequence $\mathsf{B}(k,l){\leftarrow}\bot;\mathsf{B}(k',l'){\leftarrow}\top$.

The set of *formulas* $\mathcal{F}$ is defined by the following BNF:

$$\varphi ::= q \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists\pi.\varphi$$

where $q$ ranges over $\mathbb{P}$ and $\pi$ ranges over the set of events $\Pi$. (Observe that we use $\Phi$ for Boolean formulas and $\varphi$ for formulas of DL-PA.) The formula $\exists\pi.\varphi$ reads "there is an execution of the event $\pi$ after which $\varphi$ is true". Hence $\exists\pi.\top$ has to be read "$\pi$ may occur".

The operators '?', ';', '$\cup$' and '$^{<\infty}$' are familiar from propositional dynamic logic PDL. (We could as well use the Kleene star and write $\pi^*$ instead of $\pi^{<\infty}$, as customary in PDL.) In uninterpreted PDL these operators combine abstract atomic programs, while in interpreted PDL they combine assignments of object variables to values from some domain. In contrast, atomic programs are here assignments of truth values to propositional variables, as previously studied in the dynamic epistemic logic literature [22, 8, 2].

The formula $\exists(q{\leftarrow}\top \cup q{\leftarrow}\bot).\varphi$ has the same interpretation as the quantified Boolean formula (QBF) $\exists q.\varphi$ [12]. The language of DL-PA can therefore be viewed as a generalisation of quantification over Boolean variables to complex 'quantification programs'.

The *length* of a formula $\varphi$, noted $|\varphi|$, is the number of symbols used to write down $\varphi$ — without '⟨', '⟩', dots, and parentheses —, where integers are supposed to have length 1.[3] For example

$$|\exists(q{\leftarrow}\top)^{\leq 3}.\,(q \vee r)| = 1 + |q{\leftarrow}\top| + 1 + 1 + |q \vee r| = 1 + 3 + 2 + 3 = 9.$$

The logical operators $\wedge$ and $\rightarrow$ are defined as abbreviations; for example $\varphi \rightarrow \psi$ abbreviates $\neg\varphi \vee \psi$. Moreover, $\forall\pi.\,\varphi$ abbreviates $\neg\exists\pi.\,\neg\varphi$. Hence $\forall\pi.\,\bot$ has to be read "$\pi$ cannot occur". Familiar program constructions can also be defined as abbreviations: The skip event ('nothing happens') is defined as an abbreviation of $\top?$, and if $\varphi$ then $\pi_1$ else $\pi_2$ is defined as an abbreviation of $(\varphi?;\pi_1)\cup(\neg\varphi?;\pi_2)$.

*Remark 1.* Note that could define programs $\pi^k$ as abbreviations of the sequential composition of $\pi$, $k$ times,[4] and then define $\pi^{\leq n}$ as abbreviations of $\bigcup_{0\leq k\leq n}\pi^k$. However, the expansion of these abbreviations would exponentially increase formula length. For the same reason we have avoided to introduce '$\leftrightarrow$' as an abbreviation.

We use $\tau$ as a placeholder for either $\top$ or $\bot$, and write $q{\leftarrow}\tau$ in order to talk about $q{\leftarrow}\top$ and $q{\leftarrow}\bot$ in an economic way.

### 3.2 Semantics

A *valuation* is nothing but a model of classical propositional logic, *viz.* a subset of the set of propositional variables $\mathbb{P}$.

A valuation $V$ provides truth values for propositional variables. These truth values are modified by events: to each event $\pi$ we associate a binary relation on valuations $R_\pi$ that is inductively defined as follows:

$$\begin{array}{ll}
R_{p\leftarrow\top} & = \{(V,V') \;:\; V' = V \cup \{p\}\} \\
R_{p\leftarrow\bot} & = \{(V,V') \;:\; V' = V \setminus \{p\}\} \\
R_{\pi_1;\pi_2} & = R_{\pi_1} \circ R_{\pi_2} \\
R_{\pi_1\cup\pi_2} & = R_{\pi_1} \cup R_{\pi_2} \\
R_{\varphi?} & = \{(V,V) \;:\; V \models \varphi\} \\
R_{\pi^{=n}} & = (R_\pi)^n \\
R_{\pi^{\leq n}} & = \bigcup_{0\leq m\leq n}(R_\pi)^m \\
R_{\pi^{<\infty}} & = \bigcup_{0\leq m}(R_\pi)^m
\end{array}$$

We call the valuations $V'$ such that $(V,V') \in R_\pi$ the *possible updates of $V$ by $\pi$*. Observe that $R_{\top?}$ is indeed the 'nothing happens' event relating every valuation to itself.

Then the truth conditions are the usual ones for $\top$, $\bot$, negation and disjunction, plus one for $\exists\pi.\,\varphi$ in terms of possible updates:

$$\begin{array}{ll}
V \models p & \text{iff } p \in V \\
V \models \top & \\
V \not\models \bot & \\
V \models \neg\varphi & \text{iff } V \not\models \varphi \\
V \models \varphi \vee \psi & \text{iff } V \models \varphi \text{ or } V \models \psi \\
V \models \exists\pi.\,\varphi & \text{iff there is } V' \text{ such that } VR_\pi V' \text{ and } V' \models \varphi
\end{array}$$

[3] Precisely, the length of an integer $n$ should be $\log n$. Our hypothesis is however without harm.

[4] The precise definition is inductive: $\pi^0 = \mathsf{skip}$, and $\pi^{k+1} = \pi^k;\pi$.

A valuation $V$ is a *model of* $\varphi$ if and only if $V \models \varphi$. A formula $\varphi$ is satisfiable if and only if there exists a model of $\varphi$, and $\varphi$ is valid if and only if every valuation is a model of $\varphi$.

### 3.3 Reduction axioms for the star-free fragment

The fragment of DL-PA without arbitrary iterations (called the 'star-free fragment' in PDL) can be axiomatised by means of reduction axioms. These axioms allow to eliminate all the dynamic operators from formulas. However, that elimination might result in an exponential blowup due to the operator of nondeterministic composition $\cup$. We will therefore later on characterize the complexity of validity checking by other means.

**Proposition 1.** *The following equivalences are DL-PA valid.*

$$
\begin{array}{ll}
\exists \pi^{=n}.\, \varphi & \leftrightarrow \begin{cases} \varphi & \text{if } n = 0 \\ \exists \pi^{=n-1}.\, \exists \pi.\, \varphi & \text{if } n > 0 \end{cases} \\
\exists \pi^{\leq n}.\, \varphi & \leftrightarrow \begin{cases} \varphi & \text{if } n = 0 \\ \exists \pi^{\leq n-1}.\, (\varphi \vee \exists \pi.\, \varphi) & \text{if } n > 0 \end{cases} \\
\exists \varphi?.\, \psi & \leftrightarrow \varphi \wedge \psi \\
\exists \pi_1 \cup \pi_2.\, \varphi & \leftrightarrow \exists \pi_1.\, \varphi \vee \exists \pi_2.\, \varphi \\
\exists \pi_1 ; \pi_2.\, \varphi & \leftrightarrow \exists \pi_1.\, \exists \pi_2.\, \varphi \\
\exists p{\leftarrow}\tau.\, \neg\varphi & \leftrightarrow \neg \exists p{\leftarrow}\tau.\, \varphi \\
\exists p{\leftarrow}\tau.\, (\varphi_1 \vee \varphi_2) & \leftrightarrow \exists p{\leftarrow}\tau.\, \varphi_1 \vee \exists p{\leftarrow}\tau.\, \varphi_2 \\
\exists p{\leftarrow}\tau.\, \top & \leftrightarrow \top \\
\exists p{\leftarrow}\tau.\, \bot & \leftrightarrow \bot \\
\exists p{\leftarrow}\tau.\, q & \leftrightarrow \begin{cases} \tau & \text{if } q = p \\ q & \text{if } q \neq p \end{cases}
\end{array}
$$

These equivalences provide a complete set of reduction axioms for $^\infty$-free dynamic operators $\exists\pi$. Call red the mapping on DL-PA formulas which iteratively applies the above equivalences from the left to the right, starting from one of the innermost modal operators. It allows to first eliminate complex events, then push the dynamic operators inside the formula, and finally eliminate them when facing an atomic formula. For example, consider the complex formula $\exists p{\leftarrow}\bot^2.\, (p \vee q)$. First the complex event $p{\leftarrow}\bot^2$ is eliminated, then the innermost modal operator is distributed over $\vee$ and eliminated, and finally the outermost modal operator is distributed and eliminated:

$$
\begin{array}{ll}
\exists p{\leftarrow}\bot^2.\, (p \vee q) & \leftrightarrow \exists p{\leftarrow}\bot.\, \exists p{\leftarrow}\bot.\, (p \vee q) \\
& \leftrightarrow \exists p{\leftarrow}\bot.\, (\exists p{\leftarrow}\bot.\, p \vee \exists p{\leftarrow}\bot.\, q) \\
& \leftrightarrow \exists p{\leftarrow}\bot.\, (\bot \vee q) \\
& \leftrightarrow \exists p{\leftarrow}\bot.\, q \\
& \leftrightarrow q
\end{array}
$$

We have simplified a bit between the 3rd and 4th line, replacing the subformula $\bot \vee q$ by the equivalent $q$.

**Proposition 2.** *Let $\varphi$ be a formula in the language of DL-PA without the arbitrary iteration operator $\pi^{<\infty}$. Then*

1. $\mathsf{red}(\varphi)$ *has no modal operators;*
2. $\mathsf{red}(\varphi) \leftrightarrow \varphi$ *is DL-PA valid;*
3. $\mathsf{red}(\varphi)$ *is DL-PA valid iff* $\mathsf{red}(\varphi)$ *is valid in classical propositional logic.*

Proposition 2 indicates a close relationship between star-free DL-PA and propositional logic. The merit of former over the latter is to provide a model theory and an intuitive and more succinct language. Both are valuable from a modelling perspective.

### 3.4 Complexity for the star-free fragment

Proposition 2 tells us that star-free DL-PA is not more expressive than classical propositional logic. However, reduction may exponentially increase the length of the formula because of the event operators $\cup$ and $\pi^{\leq n}$. But this is a suboptimal procedure, as we shall see now.

We first give the complexity result for model checking. The inputs of the model checking problem are a valuation $V$ and a formula $\varphi$, and the problem is to decide whether $V \models \varphi$.

**Theorem 1.** *The problem of star-free DL-PA model checking is PSPACE-complete.*
PROOF. We first establish *hardness* by reducing the problem of validity of QBFs to star-free DL-PA model checking. Consider a fully quantified Boolean formula

$$\Phi = \exists q_1 \forall q_2 \exists q_3 \ldots \exists q_{m-1} \forall q_m . \varphi$$

where $m \geq 0$ is even and where $\varphi(q_1, \ldots, q_m)$ is a propositional formula containing no variables other than $q_1, \ldots, q_m$. (The hypothesis that the number of quantifiers is even is without loss of generality: it suffices to add a dummy variable $q_m$ not occurring in $\varphi$.) We define

$$\Phi^{\mathsf{DL\text{-}PA}} = \exists \pi_1 . \forall \pi_2 . \ \ldots \exists \pi_{m-1} . \forall \pi_m . \varphi$$

where $\pi_k = q_k{\leftarrow}\top \cup q_k{\leftarrow}\bot$, for $1 \leq k \leq m$. Consider the valuation over the set $\mathbb{P} = \{q_1, \ldots, q_m\}$ of propositional variables such that, say, $V(q_i) = \mathtt{ff}$ for all $q_i$. It is readily checked that $\Phi$ is valid in Quantified Boolean Logic iff $\Phi^{\mathsf{DL\text{-}PA}}$ is true in $V$. Since both the size of $\Phi^{\mathsf{DL\text{-}PA}}$ and the size of the model are linear in the size of $\Phi$, we conclude that free-star DL-PA model checking is PSPACE-hard.

Proof of *membership* requires a recursive definition of the set of sequences of atomic events *admitted* by a complex event $\pi$.

$$\begin{aligned}
adm(p{\leftarrow}\tau) &= \{p{\leftarrow}\tau\} \\
adm(\Phi?) &= \{\Phi?\} \\
adm(\pi_1;\pi_2) &= \{\alpha_1;\alpha_2 \ : \ \alpha_1 \in adm(\pi_1) \text{ and } \alpha_2 \in adm(\pi_2)\} \\
adm(\pi_1 \cup \pi_2) &= adm(\pi_1) \cup adm(\pi_2) \\
adm(\pi^{=0}) &= \{\top?\} \\
adm(\pi^{=n+1}) &= \{\alpha_1;\alpha_2 \ : \ \alpha_1 \in adm(\pi^{=n}) \text{ and } \alpha_2 \in adm(\pi)\} \\
adm(\pi^{\leq 0}) &= \{\top?\} \\
adm(\pi^{\leq n+1}) &= adm(\pi^{\leq n}) \cup \\
&\qquad \{\alpha_1;\alpha_2 \ : \ \alpha_1 \in adm(\pi^{\leq n}) \text{ and } \alpha_2 \in adm(\pi)\}
\end{aligned}$$

The main point in the proof is that every possible update of a valuation $V$ by a complex event $\pi$ can also be reached by a sequence of atomic events that is admitted by $\pi$ and that is at most as long as $\pi$. Based on that one can prove that membership of a couple of valuations $(V, V')$ in $R_\pi$ can be decided in polynomial space. Finally one can prove that a formula $\varphi$ can be evaluated in space polynomial in the size of $\varphi$. ■

**Theorem 2.** *The problem of star-free DL-PA validity checking is PSPACE-complete.*

PROOF. Hardness can be proved by translating QBF formulas to star-free DL-PA in the same way as in Theorem 1. Membership can be proved for the star-free DL-PA satisfiability checking problem as follows: given $\varphi$ we guess a model $V$. ($V$ can be supposed to be of polynomial size because we may restrict our attention to the propositional variables occurring in $\varphi$, and neglect those that don't.) Then we check whether $V \models \varphi$, which can be done in polynomial space by mirroring the truth conditions. This shows that star-free DL-PA satisfiability can be checked in NPSPACE. Now by Savitch's theorem NPSPACE = PSPACE, and therefore star-free DL-PA satisfiability can be checked in polynomial space. It follows that the complementary star-free DL-PA validity problem can be checked in polynomial space, too. ■

The above result shows that DL-PA is *more succinct* than propositional logic: there are DL-PA formulas (and even star-free DL-PA formulas) such that every equivalent propositional formula is exponential longer.

## 4 The segregation game in DL-PA

In Section 2 we introduced Schelling's segregation game in an informal way. Let us now model it in DL-PA as a cellular automaton. We start by describing agents whose tolerance threshold is 0.

### 4.1 Propositional variables

We need three kinds of propositional variables:

| | |
|---|---|
| $\mathsf{R}(k,l)$ | "cell $(k,l)$ is red" |
| $\mathsf{B}(k,l)$ | "cell $(k,l)$ is blue" |
| $\mathsf{Done}(k,l)$ | "it was already cell $(k,l)$'s turn in the present step" |

where every $(k,l)$ is a location in the grid $[1..N] \times [1..N]$. The following abbreviations will be useful:

$$\mathsf{NB}_{<1}(k,l) \stackrel{\mathtt{def}}{=} \bigwedge_{k',l' \leq N \,:\, |k'-k|,|l'-l| \leq 1} \neg \mathsf{B}(k',l')$$
$$\mathsf{NR}_{<1}(k,l) \stackrel{\mathtt{def}}{=} \bigwedge_{k',l' \leq N \,:\, |k'-k|,|l'-l| \leq 1} \neg \mathsf{R}(k',l')$$
$$\mathsf{UnH}_{<1}(k,l) \stackrel{\mathtt{def}}{=} (\mathsf{R}(k,l) \wedge \neg \mathsf{NB}_{<1}(k,l)) \vee (\mathsf{B}(k,l) \wedge \neg \mathsf{NR}_{<1}(k,l))$$
$$\mathsf{Free}(k,l) \stackrel{\mathtt{def}}{=} \neg \mathsf{R}(k,l) \wedge \neg \mathsf{B}(k,l)$$
$$\mathsf{Segreg}_{<1} \stackrel{\mathtt{def}}{=} \neg \bigvee_{k,l} \mathsf{UnH}_{<1}(k,l)$$

$\mathsf{NB}_{<1}(k,l)$ can be read "location $(k,l)$ has no blue neighbour" and similarly for $\mathsf{NR}_{<1}(k,l)$. $\mathsf{UnH}_{<1}(i,k,l)$ can be read "location $(k,l)$ is inhabited and has some neighbour with a different colour". $\mathsf{Free}(k,l)$ can be read "location $(k,l)$ is free". Finally, the property of segregation holds —noted $\mathsf{Segreg}_{<1}$— if and only if "every inhabitant has no neighbour with a different colour".

Let us compute the length of these formulas. The length of both $\mathsf{NB}_{<1}(k,l)$ and $\mathsf{NR}_{<1}(k,l)$ is in $O(1)$, *i.e.* it is constant (the quantification over the locations $(k',l')$ being void because there are exactly eight neighbour locations of $(k,l)$). The same holds for the length of $\neg\mathsf{UnH}_{<1}(k,l)$. Finally, the length of $\mathsf{Segreg}_{<1}$ is in $O(N^2)$.

### 4.2 Describing inhabitants' moves

The move of an inhabitant from a location $(k,l)$ to the location $(k',l')$ is modelled by the swap of color between cells $(k,l)$ and $(k',l')$. It can be described by a complex event in our language of events.

$$\begin{aligned}\mathsf{move}(k,l) \stackrel{\text{def}}{=} \bigcup_{k',l'} & (\neg\mathsf{B}(k',l') \wedge \neg\mathsf{R}(k',l'))? \;; \\ & (\,(\mathsf{B}(k,l)? \;;\; \mathsf{B}(k,l){\leftarrow}\bot \;;\; \mathsf{B}(k',l'){\leftarrow}\top) \cup \\ & \;\;(\mathsf{R}(k,l)? \;;\; \mathsf{R}(k,l){\leftarrow}\bot \;;\; \mathsf{R}(k',l'){\leftarrow}\top)\,)\end{aligned}$$

Then a move of the simulation is described by a nondeterministic composition of agent moves (plus some turn taking management):

$$\begin{aligned}\mathsf{move} \stackrel{\text{def}}{=} & \Big(\textstyle\bigcup_{k,l} \neg\mathsf{Done}(k,l)?; (\mathsf{UnH}_{<1}(k,l)?; \mathsf{move}(k,l)) \cup \neg\mathsf{UnH}_{<1}(k,l)?); \mathsf{Done}(k,l){\leftarrow}\top\Big) \;; \\ & \Big(((\textstyle\bigwedge_{k,l} \mathsf{Done}(k,l))? \;; \mathsf{initialize}\;) \cup (\neg \textstyle\bigwedge_{k,l} \mathsf{Done}(k,l))?\Big)\end{aligned}$$

Using if-then-else this can be written in a way that is perhaps more understandable as:

$$\begin{aligned}\mathsf{move} \stackrel{\text{def}}{=} & \Big(\textstyle\bigcup_{k,l} \neg\mathsf{Done}(k,l)?; (\text{if } \mathsf{UnH}_{<1}(k,l) \text{ then } \mathsf{move}(k,l) \text{ else skip}) \;;\; \mathsf{Done}(k,l){\leftarrow}\top\Big) \;; \\ & \Big(\text{if } (\textstyle\bigwedge_{k,l} \mathsf{Done}(k,l)) \text{ then } \mathsf{initialize} \text{ else skip}\Big)\end{aligned}$$

The $\mathsf{initialize}$ event starts a new simulation step, setting the $\mathsf{Done}(k,l)$ variables of the inhabited cells to false: after it, none of the inhabited cells has played yet.

$$\begin{aligned}\mathsf{initialize} \stackrel{\text{def}}{=} & (\mathsf{B}(k_1,l_1) \vee \mathsf{R}(k_1,l_1))? \;; \mathsf{Done}(k_n,l_n){\leftarrow}\bot \;; \ldots \;; \\ & (\mathsf{B}(k_n,l_n) \vee \mathsf{R}(k_n,l_n))? \;; \mathsf{Done}(k_n,l_n){\leftarrow}\bot\end{aligned}$$

Nondeterministic choice of a location in the $\mathsf{move}$ formula corresponds to the scheduler in simulation platforms such as NetLogo or GAMA. The latter generates at the beginning of each step a random ordering of the set of agents and then activates the agents according to that ordering during the step. A simulation step consists in $N^2$ executions of $\mathsf{move}$.

The lengths of both $\mathsf{move}(k,l)$ and $\mathsf{initialize}$ are in $O(N^2)$. The length of $\mathsf{move}$ is in $O(N^2 \times N^2) = O(N^4)$.

### 4.3 Initial state

In order to properly analyse the segregation model, we must describe the *initial state* in our formalism:

$$\mathsf{Init} = \left(\bigwedge_{(k,l)\in J_R} \mathsf{R}(k,l)\right) \wedge \left(\bigwedge_{(k,l)\in J_B} \mathsf{B}(k,l)\right) \wedge \left(\bigwedge_{(k,l)\in J_R\cup J_B} \neg\mathsf{Done}(k,l)\right)$$

where $J_R \subseteq N^2$ is the set of red cells and $J_B \subseteq N^2$ the set of blue cells. As a cell cannot be blue and red these two sets have to be disjoint, *i.e.* we require that $J_R \cap J_B = \emptyset$. Moreover, the numbers of colored cells must be smaller than the total number of cells: $|J_R| + |J_B| \leq N^2$. Finally, note that we do not say anything about the status of uninhabited cells: we do not care whether $\mathsf{Done}(k,l)$ is true or false there. The length of $\mathsf{Init}$ is in $O(N^2)$.

### 4.4 Describing and proving properties

**Properties to be maintained (invariants).** During the execution of a simulation, several properties should be maintained and thus be always true. In the case of Schelling's game, a cell cannot be both red and blue:

$$\chi_1 = \bigwedge_{k,l} \neg\left(\mathsf{B}(k,l) \wedge \mathsf{R}(k,l)\right)$$

We need an auxiliary definition which expresses that at least $n$ individuals have property $P$:

$$\mathsf{atleast}\,(n,P) = \left(\bigvee_{(k_1,l_1),\ldots,(k_n,l_n)\,:\,(k_i,l_i)\neq(k_j,l_j)\text{ if }i\neq j} P(k_1,l_1) \wedge \ldots \wedge P(k_n,l_n)\right)$$

This allows us to state that the numbers of red and blue cells are exactly $|J_R|$ and $|J_B|$ just as in the initial state:

$$\chi_2 = \mathsf{atleast}\,(|J_R|,\mathsf{R}) \wedge \neg\mathsf{atleast}\,(|J_R|+1,\mathsf{R}) \wedge \mathsf{atleast}\,(|J_B|,\mathsf{B}) \wedge \neg\mathsf{atleast}\,(|J_B|+1,\mathsf{B})$$

Together these formulas make up the invariants of Schelling's game:

$$\mathsf{Invs} = \chi_1 \wedge \chi_2$$

The length of $\chi_1$ is in $O(N^2)$, while that of $\chi_2$ is in $O(2^{N^2})$ (which means that will be is costly to check the latter property).

The difference between $\mathsf{Init}$ and $\mathsf{Invs}$ is that while $\mathsf{Init}$ has just to be true in the initial state, the laws must be true in the initial state and in any update of the current state. In order to ensure that our modelling works properly the first things to do is to check that the invariants hold verified in the initial state and are then preserved by any sequence of events from $\mathsf{move}$. In order to prove this it suffices to prove that the formulas

$$\mathsf{Init} \rightarrow \mathsf{Invs}$$
$$\mathsf{Invs} \rightarrow [\forall\mathsf{move}]\,\mathsf{Invs}$$

are both DL-PA valid. From this it follows by standard principles of modal logic that $\mathsf{Invs} \rightarrow [\forall\mathsf{move}^n]\,\mathsf{Invs}$ and $\mathsf{Invs} \rightarrow [\forall\mathsf{move}^{\leq n}]\,\mathsf{Invs}$ are DL-PA valid.

**Properties to be achieved.** In our language we can express things such as "segregation *will always* occur after $n$ moves" ($\varphi_1$), "segregation *may* occur within $n$ moves" ($\varphi_2$), "when segregation occurs then none of the agents will move" ($\varphi_3$), etc.:

$$\varphi_1 = \forall \mathsf{move}^{=n}.\, \mathsf{Segreg}_{<1}$$
$$\varphi_2 = \exists \mathsf{move}^{\leq n}.\, \mathsf{Segreg}_{<1}$$
$$\varphi_3 = \mathsf{Segreg}_{<1} \rightarrow \forall \mathsf{move}.\, \bot$$

Other kinds of properties will be discussed in Section 5.

Given a property described by a formula $\varphi$, what we are interested in is to check whether the formula

$$\mathsf{Init} \rightarrow \varphi$$

is DL-PA valid, where $\varphi$ is one of the above properties.

It is important to note that the lengths of the formulas $\mathsf{Segreg}_{<1}$, $\neg\mathsf{UnH}_{<1}(k,l)$, Init and of the complex event move are polynomial in the domain size parameter $N$; precisely, their maximum length is in $O(N^4)$. Therefore the length of the formulas $\mathsf{Init} \rightarrow \mathsf{Invs}$, $\mathsf{Invs} \rightarrow \forall \mathsf{move}.\, \mathsf{Invs}$ and $\mathsf{Init} \rightarrow \varphi_k$ is polynomial in $N$; precisely, their length is in $O(N^4)$. The validity problem in star-free DL-PA being in PSPACE we obtain the following results.

**Proposition 3.** *The validity of* $\mathsf{Init} \rightarrow \varphi_k$*, for* $\varphi_k \in \{\varphi_1, \varphi_2, \varphi_3\}$*, can be checked in space polynomial in* $N$*.*

We note that checking whether the invariants Invs hold is more expensive because of the combinatorial explosion in the expression of the constraint that the number of red and blue agents is constant.

All our decision problems being in PSPACE, we can envisage to use existing theorem provers for PSPACE problems to check the above properties, such as provers for modal logic K, for description logic ALC, or for Quantified Boolean Formulas. This requires a polynomial transformation of the formulas to be checked into the language of the respective logic.[5]

### 4.5 Varying the inhabitants' tolerance

The inhabitants modelled here are extremely intolerant. More tolerant inhabitants can be described as follows:

$$\mathsf{NB}_{<2}(k,l) \stackrel{\mathtt{def}}{=} \bigwedge_{(k_1,l_1),(k_2,l_2)\,:\,|k_1-k|,|l_1-l|,|k_2-k|,|l_2-l|\leq 1} \neg(\mathsf{B}(k_1,l_1) \wedge \mathsf{B}(k_2,l_2))$$
$$\mathsf{NB}_{<p}(k,l) \stackrel{\mathtt{def}}{=} \bigwedge_{(k_1,l_1),\ldots,(k_p,l_p)\,:\,|k_m-k|,|l_m-l|\leq 1} \neg(\mathsf{B}(k_1,l_1) \wedge \ldots \wedge \mathsf{B}(k_p,l_p))$$

$\mathsf{NB}_{<p}(k,l)$ is read "location $(k,l)$ has less than $p$ blue neighbours". $\mathsf{NR}_{<p}(k,l)$ is defined accordingly. $\mathsf{UnH}_{<1}(k,l)$ and $\mathsf{Segreg}_{<1}$ can be generalised to $\mathsf{UnH}_{<p}(k,l)$ and $\mathsf{Segreg}_{<p}$ in the obvious way. One may also stipulate that an agent is happy if the percentage of agents in his neighbourhood with a colour different from his is below some threshold.

[5] While we know that such a transformation exists because all these problems are in the same complexity class, it remains to find an elegant such transformation.

For the same reason as for $\mathsf{NB}_{<1}$, the length of $\mathsf{NB}_{<2}$ and of any $\mathsf{NB}_{<p}$ ($p \leq 8$) is in $O(1)$. It follows that the length of $\mathsf{UnH}_{<2}(k,l)$ (and $\mathsf{UnH}_{<p}(k,l)$) is still in $O(N^2)$, that of $\mathsf{Segreg}_{<2}$ (and $\mathsf{Segreg}_{<p}$) is still in $O(N^2)$, and that of $\mathsf{move}$ is still in $O(N^4)$. So complexity does not increase for these generalisations and just as in Section 4.4 such properties can still be checked in polynomial space.

## 5 More expressive languages

We now discuss some generalisations of our logic that allow to naturally express other properties one would like to check in simulations.

Let us introduce two new modal operators $\geq k\ \pi$ and $\geq \frac{1}{2}\ \pi$, where $\geq k\ \pi.\varphi$ reads "$\varphi$ is true in at least $k$ of the possible updates by $\pi$" and $\geq \frac{1}{2}\ \pi.\varphi$ reads "$\varphi$ is true in most of the states after $\pi$". So the formula $\exists\pi.\varphi$ of Section 3 is nothing but $\geq 1\ \pi.\varphi$.

$$V \models \geq k\ \pi.\varphi \text{ iff } |\{V' : (V,V') \in R_\pi \text{ and } V' \models \varphi\}| \geq k$$
$$V \models > \tfrac{1}{2}\ \pi.\varphi \text{ iff } |\{V' : VR_\pi V' \ \&\ V' \models \varphi\}| > |\{V' : VR_\pi V' \ \&\ V' \not\models \varphi\}|$$

This allows to formulate interesting properties of segregation game such as:

- "segregation will occur within $n$ moves *at least* $k$ times" ($\psi_1$);
- "segregation will occur within $n$ moves *exactly* $k$ times" ($\psi_2$);
- "segregation will occur after $n$ moves in most of the cases" ($\psi_3$).

In formulas:

$$\psi_1 = \geq k\ \mathsf{move}^{\leq n}.\mathsf{Segreg}_{<p}$$
$$\psi_2 = \geq k\ \mathsf{move}^{\leq n}.\mathsf{Segreg}_{<p} \wedge \neg \geq k{+}1\ \mathsf{move}^{\leq n}.\mathsf{Segreg}_{<p}$$
$$\psi_3 = > \tfrac{1}{2}\ \mathsf{move}^{=n}.\mathsf{Segreg}_{<p}$$

Model checking requires some more bookkeeping in order to count valuations, but can still be done in polynomial space. We obtain a PSPACE completeness result for the validity checking problem by using Savitch's theorem in the same way as we did in the proof of Theorem 2.

## 6 Conclusion

In this paper we have shown how to do social simulation in a dynamic logic with assignments, tests, sequential and nondeterministic composition, and bounded and non-bounded iteration. In Section 5 we have shown that our logic allows to study interesting properties of segregation game. For instance, it allows to test whether, given a certain initial configuration of the grid, segregation will emerge at some point in the future at least $k$ times (or exactly $k$ times). In order to test such properties by standard simulation methods it would be necessary to run several computer simulations and to then perform a statistical analysis of the results that have been observed. This allows then to estimate the frequency with which segregation emerges at some point of the simulation. In this sense, the approach proposed in this work offers a novel method for social simulation studies and analysis. We have started to implement the segregation game in QBF provers [6].

In this paper, we focused on a cellular automaton version of the segregation model. We did so in order to reduce the length of our formulas describing the segregation game. We present now a sketch of formalization of the agent-based model in order to illustrate the possible use of our logic not only in cellular automaton-based simulations but also in agent-based simulations. In the agent-based version of the model, every inhabitant would be represented by an individual (typically noted $i, j, \ldots$) who is situated on a grid cell that is described by its coordinates $(k, l)$. Each of these agents can move on the grid from one cell to another one. Each agent is characterized by his colour (*e.g.* red or blue) and his location on the grid. We thus have to adapt the propositional variables in order to take into account this new representation. We do so by introducing $\mathsf{At}(i, k, l)$ variables expressing that agent $i$ is at location $(k, l)$. The variables $\mathsf{R}(k, l)$ and $\mathsf{Done}(k, l)$ will be changed to $\mathsf{R}(i)$ and $\mathsf{Done}(i)$, because agents and no more cells will have a color and perform actions. The simulation will stop when every agent is happy ($\mathsf{UnH}_{<1}(k, l)$ will thus become $\mathsf{UnH}_{<1}(i, k, l)$). This new formalization will induce a noticeable increase of complexity of some of our formulas; for example, the length of the formula $\mathsf{move}$ is currently in $O(N^4)$, while it will be in $O(N^{22})$ in the agent version when we take the parameter $p = 8$. Nevertheless, the possibility to represent an agent-based simulation model in our logic opens us much more possibilities of interesting applications.

Instead of our logic we could also have used other logical approaches to reasoning about actions such as the Situation Calculus [16], the Fluent Calculus [20], or the Event Calculus [18]. However, while these formalisms allow to represent more or less the same things, their mathematical analysis is less developed: while there are some decidability results, there are no complexity results that could be compared to the PSPACE completeness result for our logic.

As we have said in the introduction the kind of properties we want to prove can be viewed as finite horizon planning problems. To witness, the way we prove our PSPACE complexity results matches *e.g.* the Chapman's TWEAK [4]. We could therefore have used existing finite horizon planners in order to prove properties of simulations. It has to be noted that planners typically build plans, while we are only interested in proving plan existence. However, it is an interesting research avenue to exploit a possible convergence of the fields of simulation and planning.

## 7 Acknowledgements

We wish to thank Philippe Balbiani and Charles Cultien for their useful comments on the complexity aspects of DL-PA and on the logical modeling of the segregation game. This work was partially supported by the French RTRA STAE project MAELIA.

## References

1. F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *Description Logic Handbook.* Cambridge University Press, 2003.
2. J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(204):1620–1662, 2006.

3. T. Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69:165–204, 1994.
4. D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377, 1987.
5. R. Conte and M. Paolucci. Responsibility for societies of agents. *Journal of Artificial Societies and Social Simulation*, 7(4), 2004.
6. C. Cultien. Implementing dynamic logic of propositional assignments in a QBF solver. Master's thesis, Université de Toulouse, sep 2011.
7. F. Dignum, B. Edmonds, and L. Sonenberg. Editorial: The use of logic in agent-based social simulation. *Journal of Artificial Societies and Social Simulation*, 7(4), 2004.
8. H. P. van Ditmarsch, W. van der Hoek, and B. Kooi. Dynamic epistemic logic with assignment. In *Proceedings of AAMAS'05*, pages 141–148. ACM Press, 2005.
9. B. Edmonds. How formal logic can fail to be useful for modelling or designing MAS. In G. Lindeman, editor, *Proceedings of the International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA'02)*, LNAI, pages 1–15. Springer-Verlag, 2004.
10. M. Fasli. Formal systems and agent-based social simulation equals null? *Journal of Artificial Societies and Social Simulation*, 7(4), 2004.
11. M. Fattorosi-Barnaba and F. de Caro. Graded modalities I. *Studia Logica*, 44:197–221, 1985.
12. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman Co., 1979.
13. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, 2000.
14. I. Horrocks. Using an expressive description logic: Fact or fiction? In *Proceedings of KR'98*, pages 636–649, 1998.
15. H. A. Kautz and B. Selman. Planning as satisfiability. In *Proceedings of ECAI'92*, pages 359–363, 1992.
16. R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.
17. T. C. Schelling. Dynamic Models of Segregation. *Journal of Mathematical Sociology*, 1:143–186, 1971.
18. M. Shanahan. *Solving the frame problem: a mathematical investigation of the common sense law of inertia*. MIT Press, 1997.
19. P. Taillandier, A. Drogoul, D.A. Vo, and E. Amouroux. GAMA: a simulation platform that integrates geographical information data, agent-based modeling and multi-scale control. In *Proceedings of PRIMA'10*, 2010.
20. M. Thielscher. The logic of dynamic systems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1956–1962, Montreal, Canada, 1995.
21. W. van der Hoek. On the semantics of graded modalities. *Journal of Applied Non-Classical Logics*, 2(1), 1992.
22. J. van Eijck. Making things happen. *Studia Logica*, 66(1):41–58, 2000.
23. U. Wilensky. Netlogo segregation model. Technical report, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1997.
24. U. Wilensky. Netlogo. Technical report, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999.