stigLD: Stigmergic Coordination of Linked Data Agents

René Schubotz(✉), Torsten Spieldenner, and Melvin Chelli

German Research Center for Artificial Intelligence,
Saarland Informatics Campus D 3 2, Saarbrücken, Germany
rene.schubotz@dfki.de

Abstract. While current Semantic Web technologies are well-suited for data publication and integration, the design and deployment of dynamic, autonomous and long-lived multi-agent systems (MAS) on the Web is still in its infancy. Following the vision of hypermedia MAS and Linked Systems, we propose to use a value-passing fragment of Milner's Calculus to formally specify the generic hypermedia-driven behaviour of Linked Data agents and the Web as their embedding environment. We are specifically interested in agent coordination mechanisms based on stigmergic principles. When considering transient marker-based stigmergy, we identify the necessity of generating server-side effects during the handling of safe and idempotent agent-initiated resource requests. This design choice is oftentimes contested with an imprecise interpretation of HTTP semantics, or with rejecting environments as first-class abstractions in MAS. Based on our observations, we present a domain model and a SPARQL function library facilitating the design and implementation of stigmergic coordination between Linked Data agents on the Web. We demonstrate the efficacy our modeling approach in a Make-to-Order fulfilment scenario involving transient stigmergy and negative feedback.

Keywords: Linked Data · Semantic Web · Multi-agent systems · Stigmergy · Nature inspired algorithm · RDF · SPARQL

1 Introduction

Hypermedia multi-agent systems [4,6], sometimes also referred to as Linked Systems [20], are receiving increasing research attention. The hypothesis is that the Web provides a scalable and distributed hypermedia environment that embedded agents can use to uniformly discover and interact with other agents and artifacts. Following a set of design principles very much aligned with REST and Linked Data best practices [2], the design and deployment of world-wide and long-lived hypermedia MASs with enhanced scalability and evolvability is aspired. In this context, we are specifically interested in stigmergic coordination principles for hypermedia MASs. The concept of stigmergy [22] provides an indirect and mediated feedback mechanism between agents, and enables complex,

L. Pan et al. (Eds.): BIC-TA 2021, CCIS 1566, pp. 174–190, 2022.
https://doi.org/10.1007/978-981-19-1253-5_13

coordinated activity without any need for planning and control, direct communication, simultaneous presence or mutual awareness. A crucial part of a stigmergic system is its stigmergic environment [35] given that "it is its mediating function that underlies the power of stigmergy" [23]. Accounting for the importance of distributed hypermedia environments as first-class abstractions in hypermedia MASs and the environment's pivotal role in stigmergic systems, we examine the use of hypermedia-enabled Linked Data as a general stigmergic environment.

We briefly present core concepts and variations of stigmergic systems and summarise existing literature relevant to our work in Sect. 2. Next in Sect. 3, we propose to use a value-passing fragment of Milner's Calculus to formally specify generic, hypermedia-driven Linked Data agents and the Web as their embedding environment. We composed Linked Data agents and their environment into a Linked System (or equivalently a hypermedia MAS). Based on this formalism, we consider transient marker-based stigmergy as coordination mechanism between Linked Data agents in Sect. 4. We identify the necessity of generating server-side effects during the handling of safe and idempotent agent-initiated requests, and present a domain model and a SPARQL function library facilitating the design and implementation of stigmergic environments on the Web. Section 5 illustrates and evaluates our approach in a Make-to-Order fulfilment scenario involving transient stigmergy and negative feedback. We conclude and point out future work in Sect. 6.

2 Varieties of Stigmergy and Related Work

In collective stigmergic systems, groups of *agents* perform work by executing *actions* within their environment [23]. An action is considered a causal process that produces a change in the environment. Agents choose actions based on condition-action rules, and perform an action as soon as its condition is found to be met. Conditions are typically based on environmental states as *perceived* by the agent. Examples from nature are the presence of specific (food) resources, semiochemical traces, progress in building nest structures, etc. Which actions an agent can perform, how the agent will perform them, and which condition-action rules an agent will follow, is considered the agent's *competence* [25]. The part of the environment that undergoes changes as a result of executing an action, and the state of which is perceived to incite further actions, is called the *medium*. Each action produces, either as byproduct of an action, or the deliberate goal of the action itself, a *stigma* in the medium. Consequently, the behaviour of agents in a collective stigmergic system can be understood as a cycle of executing actions based on existing stigmata, and as result, leaving stigmata that stimulate or inhibit future actions (see Fig. 1). In essence, stigmata work as indirect communication mechanism between agents [37], potentially leading to coordination between agents, and,

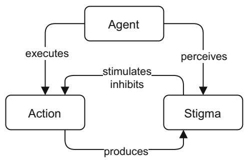

Fig. 1. Stigmergic feedback loop

ideally, a self-organising behaviour of the entire system [22–24]. Based on these core concepts, i.e. *action*, *medium* and *stigma*, stigmergic systems can be further classified [23]. In *sematectonic* stigmergy, a stigma is a perceivable modification of the environment as result of work that was carried out by the agent, e.g. giving some new shape to a working material, or re-arranging order of objects in the world. In *marker-based* stigmergy, stigmata are markers, e.g. semiochemicals, that are specifically added to the environment as means for indirect communication between agents. When perceiving stigmata, agents may choose their actions based on the mere existence of a stigma in the medium (*qualitative stigmergy*), or also take into account quantities, like semiochemcial concentration levels, number of stigmata left, etc. (*quantitative stigmergy*). Moreover, stigmata present in the medium may stay until actively being removed by an agent (*persistent stigmata*) or until dissipated over time due to agent-less processes (*transient stigmata*).

Since the concept of stigmergy was coined as inherent underlying principle of coordination found in nature, it has faced a history of thorough research [36]. There is a profound understanding of the many variations of stigmergic systems, and how these are suited to model and implement efficient, flexible, and scalable algorithms for AI-based coordination and optimization [7,23,24].

Stigmergy is recognized as suitable underlying principle for multi-agent systems [17,18,37,38] and is applied in a variety of practical domains, e.g. digital manufacturing [39], robotics [27,30] or public transport [1,29].

Stigmergic systems can be considered a variation of *situated agent systems*, in which the interaction of agents with their environment is reduced to direct reaction based on perception, rather than complex knowledge processing and inference [41–43]. Principles in these systems were also developed around an indirect, influence-based interaction mechanism between agents and their environment as chosen for our proposed stigmergic system [13].

Web technologies have been found a suitable basis for implementation of multi agent systems [5,6,26,28]. Meanwhile, it came to attention that stigmergic principles are the underlying concept of many applications in the World Wide Web [8] including coordination in Web-based IoT systems [33].

Self-organizing multi agent systems and agent systems that rely on stigmergy as coordination mechanism have been exhaustively reviewed in [3]. This review concludes that a common understanding of such systems is widely lacking, and suggests a generic domain model to describe self-organizing system. From the review, we conclude additionally that the interaction between agents and environment is often described only vaguely, and is generally underspecified. As a solution, we provide in this paper a formal and generic specification of hypermedia driven agents and the respective agent-server interaction for stigmergic systems.

3 Process Algebra, Agents and Linked Systems

In what follows, we recap the syntax and semantics of a value-passing fragment of Milner's Calculus of Communicating Systems (CCS) [31,32]. This process algebra allows us to (i) specify the notion of Linked Data servers, (ii) formally model the *generic* hypermedia-driven behaviour of *Linked Data agents*, and (iii) compose a collection of Linked Data agent and server processes into a concurrent system that is denoted as a *Linked System* [20] or a hypermedia MAS [6].

3.1 Theoretical Setting: CCS with Value-Passing

Let $\mathcal{A}$ be a set of channel names; $\bar{\mathcal{A}} = \{\bar{a} \mid a \in \mathcal{A}\}$ be the set of co-names; $Act = \mathcal{A} \cup \bar{\mathcal{A}} \cup \{\tau\}$ be the set of actions where τ is the silent action; and $\mathcal{K}$ be a set of process identifiers.

The set $\mathcal{P}$ of all *process expressions* is the set of all terms generated by the right-hand side abstract syntax. Here, $\mathbf{0}$ is the atomic inactive process; $K \in \mathcal{K}$ is a process identifier; $\alpha \in Act$; $\vec{x} = (x_1, \ldots, x_n)$ is a n-dimensional vector of variables; $P_{1 \le i \le 2} \in \mathcal{P}$ are process expressions; and e is a Boolean expression.

$$
\begin{array}{rcll}
P & := & \mathbf{0} & \text{(inaction)} \\
 & \mid & K & \text{(process labelling)} \\
 & \mid & \alpha.P & \text{(prefixing)} \\
 & \mid & \alpha(\vec{x}).P & \text{(value passing)} \\
 & \mid & P_1 + P_2 & \text{(choice)} \\
 & \mid & P_1 \parallel P_2 & \text{(parallel composition)} \\
 & \mid & \textbf{if } e \textbf{ then } P_1 \textbf{ else } P_2 & \text{(conditional)}
\end{array}
$$

A *process definition* is an equation system of the form $(K_{1 \le i \le k} = P_{1 \le i \le k})$ where $P_{1 \le i \le k} \subset \mathcal{P}$ is a set of process expression with process identifiers from $K_{1 \le i \le k} \subset \mathcal{K}$. Each process definition determines an Act-labelled transition system whose transitions can be inferred from the following Structural Operational Semantics rules

$$
\frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \frac{P \xrightarrow{\alpha} P' \quad (K = P)}{K \xrightarrow{\alpha} P'} \qquad \frac{P \xrightarrow{\alpha} P'}{(P + Q) \xrightarrow{\alpha} P'} \qquad \frac{Q \xrightarrow{\alpha} Q'}{(P + Q) \xrightarrow{\alpha} Q'}
$$

$$
\frac{P \xrightarrow{\alpha} P'}{(P \parallel Q) \xrightarrow{\alpha} (P' \parallel Q)} \qquad \frac{Q \xrightarrow{\alpha} Q'}{(P \parallel Q) \xrightarrow{\alpha} (P \parallel Q')} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{(P \parallel Q) \xrightarrow{\tau} (P' \parallel Q')}
$$

$$
\frac{}{\bar{a}(\vec{x}).P \xrightarrow{\bar{a}(\vec{v})} P} \qquad \frac{}{a(\vec{x}).P \xrightarrow{a(\vec{v})} P[^{v_1}/_{x_1}, \ldots, ^{v_n}/_{x_n}]} \qquad \frac{P \xrightarrow{\bar{a}(\vec{v})} P' \quad Q \xrightarrow{a(\vec{v})} Q'}{(P \parallel Q) \xrightarrow{\tau} (P' \parallel Q')}
$$

$$
\frac{P \xrightarrow{\alpha} P'}{\textbf{if } \mathit{true} \textbf{ then } P \textbf{ else } Q \xrightarrow{\alpha} P'} \qquad \frac{Q \xrightarrow{\alpha} Q'}{\textbf{if } \mathit{false} \textbf{ then } P \xrightarrow{\alpha} Q' \textbf{ else } Q}
$$

where $P, P', Q, Q' \in \mathcal{P}$ are process expressions; $K \in \mathcal{K}$ is a process identifier; $\alpha \in Act$; $\vec{x} = (x_1, \ldots, x_n)$; $a, \bar{a} \in \mathcal{A} \cup \bar{\mathcal{A}}$; $P[^v/_x]$ is the process expression obtained from P by substituting a data value v for all occurrences of x.

3.2 Linked Data Servers, Agents and Linked Systems

Let $\mathbf{I}$, $\mathbf{L}$ and $\mathbf{B}$ be pairwise disjoint sets of resource identifiers, literals and blank nodes, respectively. The set of all *RDF triples* is $\mathcal{T} = (\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L})$; a *RDF graph* $G \subset \mathcal{T}$ is a finite set of RDF triples. Given a formal RDF query language Q, we define the *query answering* functions $\mathtt{ans} : Q \times 2^{\mathcal{T}} \rightarrow 2^{\mathcal{T}}$, $\mathtt{ask} : Q \times 2^{\mathcal{T}} \rightarrow \mathbb{B}$, $\mathtt{sel} : Q \times 2^{\mathcal{T}} \rightarrow 2^{\mathbf{I}}$ and $\mathtt{descr} : \mathbf{I} \times 2^{\mathcal{T}} \rightarrow 2^{\mathcal{T}}$.

A *resource structure* is a tuple $(\mathbf{I}, R, \eta, \mathtt{OPS}, \mathtt{RET})$ where $\mathbf{I}$ is given as above; $R \subset \mathbf{I}$ is a finite set of root identifiers; $\eta : \mathbf{I} \rightarrow \mathbb{N}$ is a function that maps resource identifier i to its origin server $\mathtt{SERVER}_{\eta(i)}$; $\mathtt{OPS} = \{\mathtt{GET}, \mathtt{PUT}, \mathtt{POST}, \mathtt{DEL}\}$ is a set of method names; and $\mathtt{RET} = \{\mathtt{OK}, \mathtt{ERR}\}$ is a set of return codes.

We now fix a set of channel names as $\mathcal{A} = \{req_i, res_i \mid i \in \mathbb{N}\}$, and give CCS-style process specifications of *Linked Data servers* as well as *Linked Data agents* defined over the given resource structure $(\mathbf{I}, R, \eta, \mathtt{OPS}, \mathtt{RET})$.

Linked Data Servers. We conceive a Linked Data server $\mathtt{SERVER}_k$ as a reactive component that maintains an RDF graph G. It receives requests to perform a CRUD operation $op \in \mathtt{OPS}$ on a resource i via channel req_k

$$\mathtt{SERVER}_k(G) = req_k(op, i, G').\mathtt{PROC}_k(op, i, G', G)$$

where $G' \subset \mathcal{T}$ is a (potentially empty) request body. The server employs a constrained set of operations to process client-initiated requests for access and manipulation of the server-maintained RDF graph G

$$\begin{aligned}
\mathtt{PROC}_k(\mathtt{GET}, i, G', G) &= \mathtt{RESP}_k(\mathtt{OK}, (\emptyset, \mathtt{descr}(i, G)), G) \;+\; \mathtt{RESP}_k(\mathtt{ERR}, (\emptyset, \emptyset), G)\\
\mathtt{PROC}_k(\mathtt{PUT}, i, G', G) &= \mathtt{RESP}_k(\mathtt{OK}, (\emptyset, \emptyset), (G \setminus \mathtt{descr}(i, G)) \cup G') \;+\; \mathtt{RESP}_k(\mathtt{ERR}, (\emptyset, \emptyset), G)\\
\mathtt{PROC}_k(\mathtt{POST}, i, G', G) &= \mathtt{RESP}_k(\mathtt{OK}, (\{i'\}, \emptyset), G \cup G') \;+\; \mathtt{RESP}_k(\mathtt{ERR}, (\emptyset, \emptyset), G)\\
\mathtt{PROC}_k(\mathtt{DEL}, i, G', G) &= \mathtt{RESP}_k(\mathtt{OK}, (\{i\}, \emptyset), G \setminus \mathtt{descr}(i, G)) \;+\; \mathtt{RESP}_k(\mathtt{ERR}, (\emptyset, \emptyset), G)
\end{aligned}$$

where $i' \in \mathbf{I}$ is a “fresh” IRI with $\eta(i') = k$. The server responds to requests via channel $\overline{res}_k$

$$\mathtt{RESP}_k(rc, rval, G) = \overline{res}_k(rc, rval).\mathtt{SERVER}_k(G)$$

with return code $rc \in \mathtt{RET}$ and with a linkset and response graph in $rval \in (2^{\mathbf{I}} \times 2^{\mathcal{T}})$.

Tropistic Linked Data Agents. We specify a *tropistic* [16, section 13.1] Linked Data agent $\mathtt{AGENT}_k$ as an active component

$$\mathtt{AGENT}_k = \mathtt{PERC}_k(i \in R, G = \emptyset, L = \{i\})$$

being initially situated at a resource $i \in R$ without a-priori agent knowledge $(G = \emptyset)$ and a linkset $L = \{i\}$ restricted to i. Our specification of $\mathtt{AGENT}_k$ puts emphasis on a direct response to its perceptions and favours to employ *situated* *perceptions* [34] of the environment as the basis for deciding which action to perform next. We model situated perception in CCS-style as

$$\texttt{PERC}_k(i, G, L) = \overline{req}_{\eta(j)}(\texttt{GET}, j, \emptyset).res_{\eta(j)}(rc, (L', G')). \\ \Big(\texttt{PERC}_k(i, G'', L'') + \texttt{REACT}_k(i, G'', L'')\Big) \tag{1}$$

where $\texttt{AGENT}_k$ - while being situated at i - will at first issue a GET request for a resource j in its current linkset L via channel $\overline{req}_{\eta(j)}$ and then awaits the server's response via channel $res_{\eta(j)}$ with return code $rc \in \texttt{RET}$, response linkset $L' \subset \mathbf{I}$ and response graph in $G' \in \mathcal{T}$. Subsequently, the agent executes (i) a *perceptional query* $q_{\texttt{PERC}_k}$ over G' in order to update its situational knowledge to

$$G'' = G \cup \texttt{ans}(q_{\texttt{PERC}_k}, G')$$

as well as (ii) a *navigational query* $q_{\texttt{NAV}_k}$ over its updated knowledge graph in order to update its linkset to

$$L'' = L \cup L' \cup \texttt{sel}(q_{\texttt{NAV}_k}, G''))$$

On the basis of G'' and L'', $\texttt{AGENT}_k$ chooses to either recurse into its situated perception process $\texttt{PERC}_k(i, G'', L'')$ or to enter the process $\texttt{REACT}_k(i, G'', L'')$ in order to select an action on the basis of a local, short-time view of its environment. An action selected only on the basis of a situated perception is called a *reaction*.

We model the process of selecting reactions in the following way

$$\texttt{REACT}_k(i, G, L) = \texttt{PERC}_k(j \in L, \emptyset, \{j\}) + \\ \sum_{m \in \texttt{OPS} \setminus \{\texttt{GET}\}} \Big(\textbf{if } \texttt{ask}(\widehat{q}_{\texttt{m}_k}, G, L) \textbf{ then } \texttt{m}_k(i, G, L) \textbf{ else } \texttt{REACT}_k(i, G, L)\Big) \tag{2}$$

In essence, an agent may choose to either

(i) re-situate and perform situated perception of resource $j \in L, j \neq i$ with the implication that its situational knowledge and linkset will be reset; hence it does neither maintain a long-term internal model of its environment nor pursues explicit goals;

(ii) request the execution of operation $m \in \texttt{OPS} \setminus \{\texttt{GET}\}$ against resource i given that the *conditional query* $\widehat{q}_{\texttt{m}_k}$ over its knowledge graph G holds; possible instantiations of $\texttt{m}_k(i, , L)$ are given by

$$\texttt{PUT}_k(i, G, L) = \overline{req}_{\eta(i)}(\texttt{PUT}, i, \texttt{ans}(q_{\texttt{PUT}_k}, G)).res_{\eta(i)}(rc, (\emptyset, \emptyset)).\texttt{REACT}_k(i, G, L)$$

$$\texttt{POST}_k(i, G, L) = \overline{req}_{\eta(i)}(\texttt{POST}, i, \texttt{ans}(q_{\texttt{POST}_k}, G)).res_{\eta(i)}(rc, (L', \emptyset)).\texttt{REACT}_k(i, G, L \cup L')$$

$$\texttt{DEL}_k(i, G, L) = \overline{req}_{\eta(i)}(\texttt{DEL}, i, \emptyset).res_{\eta(i)}(rc, (L', \emptyset)).\texttt{REACT}_k(j \in L \setminus L', G, L \setminus L')$$

where $\texttt{ans}(q_{\texttt{m}_k}, G)$ is the result graph of executing an *effectual query* $q_{\texttt{m}_k}$ over the agent's knowledge graph G with $m \in \{\texttt{PUT}, \texttt{POST}\}$.

Given the formal notation of Linked Data servers and agents, we can now focus on composing a collection of Linked Data agent and server processes into a concurrent system that is denoted as a hypermedia MAS [6] or a *Linked System* [20].

Linked Systems. A *Linked System* [20] is the parallel composition

$$\texttt{LINKED-SYSTEM} = (\texttt{AGENTS} \parallel \texttt{ENVIRONMENT})$$

with $\texttt{AGENTS} = (\texttt{AGENT}_1 \parallel \cdots \parallel \texttt{AGENT}_m)$and $\texttt{ENVIRONMENT} = (\texttt{SERVER}_1 \parallel \cdots \parallel \texttt{SERVER}_n)$for a collection of Linked Data agents $\texttt{AGENT}_{1 \leq k \leq m}$ and Linked Data servers $\texttt{SERVER}_{1 \leq k \leq n}$ respectively. All direct interaction within LINKED-SYSTEM is between agent and server processes.

The *state space* of LINKED-SYSTEM is given by the nodes of an *Act*-labelled transition system whose transitions can be inferred from the Structural Operational Semantics rules given in Sect. 3.1.

A *computation* is an alternating sequence of global states and actions, where an *action* is either a communication between an agent and a server, or an internal process transition. A computation of a Linked System induces an *interaction sequence* given by the sequence of actions along that computation.

3.3 Synthesis

With the notions of Linked Data servers, tropistic Linked Data agents, and finally Linked Systems as defined above, the resulting value-passing CCS fragment enables us to formally specify the generic hypermedia-driven behaviour of tropistic Linked Data agents. We would like to emphasise the fact that the general behaviors as described by the CCS fragment are generic and independent of the scenarios in which they are applied. Domain- or application-specific behaviors of agents and systems are entirely encoded in terms of the queries that are evaluated as part of the different processes. For these, we identified four different type of queries:

(i) *Perceptional queries* specify the subsets of the environment representation relevant to the agent.
(ii) *Navigational queries* constrain the agent navigation with respect to such relevant subsets of the environment.
(iii) *Conditional queries* guard the selection of particular reactions.
(iv) *Effectual queries* describe how the agent intends to manipulate a given resource.

The per se generic framework can be applied to different scenarios by supplying respective specific queries. In the following section, we will extend Linked Systems to support stigmergy by an additional class of queries: *evolutional queries* that drive the dynamics of the underlying ENVIRONMENT.

4 Stigmergy in Linked Systems

A LINKED-SYSTEM as specified previously provides an indirect, mediated mechanism of coordination between AGENTS. It therefore enables the realisation of sematectonic and *persistent* marker-based stigmergy. However, when considering some of the prime examples of stigmergy, e.g. ant colony optimization [9–12] and termite colony optimisation methods [21], it becomes apparent that a purely reactive ENVIRONMENT is insufficient for the implementation of *transient marker-based stigmergic* mechanisms.

In fact, a stigmergic environment typically demonstrates some immanent dynamics that may modify the environment's state independent of any agent's actions [23, p. 24]. These endogenous dynamics, e.g. diffusion, evaporation, dissipation, atrophy or erosion of stigmata, constitute a crucial component of transient marker-based stigmergic systems ([40], cf. Fig. 2), and more importantly, they are *not* subjected to agent-driven processes. 5 We call the part of a stigmergic environment that, in addition to being malleable and perceivable by all agents under coordination, *actively* drives the evolution of such agent-less dynamic processes a *stigmergic medium*.

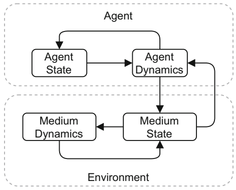

Fig. 2. Stigmergic system components

Taking into account the notion of a stigmergic medium, we define a *stigmergic Linked System* as the parallel composition

$$\texttt{STIGMERGIC-LINKED-SYSTEM} = (\texttt{AGENTS} \parallel (\texttt{MEDIUM} \parallel \texttt{ENVIRONMENT}))$$

where the stigmergic $\texttt{MEDIUM} = \texttt{MEDIUM}_1 \parallel \cdots \parallel \texttt{MEDIUM}_l$ relates to the parallel composition of a collection of *extended* LD server components.

A $\texttt{MEDIUM}_k$ component is a Linked Data server that offers a constrained set of operations to access and manipulate server-provided resource states, but *in addition*, generates server-side side-effects[1]

$$\begin{aligned}
\texttt{MEDIUM}_k(G) &= req(op, i, G').\texttt{PROC}_k(op, i, G', G)) \\
\texttt{RESP}_k(rc, rval, G) &= \overline{res}(rc, rval).\texttt{MEDIUM}_k(G) \\
\texttt{PROC}_k(\texttt{GET}, i, G', G) &= \texttt{EVOLVE}_k(i, G)
\end{aligned}$$

as evolution $\texttt{EVOLVE}_k(i, G)$ of the environment during the handling of safe and idempotent agent-initiated resource request. The generation of such side-effects is subjected to an *internal* process

$$\texttt{EVOLVE}_k(i, G) = \texttt{RESP}(\texttt{OK}, (\emptyset, \texttt{descr}(i, G')), G'') + \texttt{RESP}_k(\texttt{ERR}, (\emptyset, \emptyset), G) \tag{3}$$

where the result of executing an *evolutional query* $q_{\texttt{EVO}_k}$ over a given RDF graph G is given by $G' = \texttt{ans}(q_{\texttt{EVO}_k}, G)$ and the server state after an evolutional state update is $G'' = G \setminus \texttt{descr}(i, G) \cup \texttt{descr}(i, G')$. Executing an evolutional query drives the endogenous dynamics of $\texttt{MEDIUM}_k$ over time, e.g. diffusion and evaporation of semiochemicals, irrespectively of *agent-initiated requests for* resource state *change*.

Next, we address the definition of evolutional queries; towards this end, we introduce the stigLD domain model and the stigFN SPARQL function library.

1 We emphasise that this conception is not in violation with HTTP semantics [14, sections4.2.1,4.2.2] [15].

4.1 stigLD: A Domain Model for Stigmergic Linked Systems

Our domain model (cf. Fig. 3) defines four basic concepts: `stig:Medium`, `stig:Law`, `stig:Topos` and `stig:Stigma`.

A `stig:Medium` instance is a resource that allows for interaction between different actions, and therefore, it enables the stigmergic coordination between agents performing such actions. In order to fulfil its "mediating function that underlies the true power of stigmergy" [23], a `stig:Medium` must be similarly perceivable and malleable by all agents under stigmergic coordination. A `stig:Medium` is considered a part of a larger environment, and it undergoes changes only through agents' actions or through a set of `stig:Law` governing its endogenous dynamics.

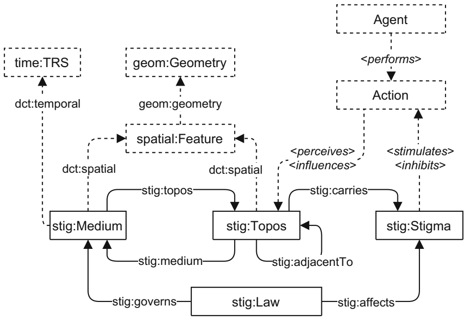

Fig. 3. stigLD domain model

A `stig:Medium` may optionally detail on its spatio-temporal characteristics[2], however, it must introduce a structure of interconnected `stig:Topos` instances in which an agent navigates, experiences situated perception and exerts situated behaviour.

A `stig:Topos` resource is the fundamental structural element of a `stig:Medium` and carries a potentially empty set of `stig:Stigma` instances. It has a potentially empty set of directed connections to other `stig:Topos` instances within the same `stig:Medium` instance. Furthermore, a `stig:Topos` may be identified with any domain- or application-specific resource using an `owl:sameAs` link and optionally detail on its spatial characteristics. An agent situated in a specific `stig:Topos` partially perceives the medium state and may try to influence the medium as a result of its action.

A `stig:Stigma` is a perceivable change made in a `stig:Medium` by an agent's action. The perception of a `stig:Stigma` may stimulate (or inhibit) the performance of a subsequent action, i.e. the presence of a `stig:Stigma` makes the performance of this action more (or less) likely. Hence, actions stimulate (or inhibit) their own continued execution via the intermediary of `stig:Stigma` (cf. Fig. 1).

A `stig:Law` describes the spatio-temporal evolution of stigmata within the medium. For this, a `stig:Law` describes itself in terms of its specific effect, e.g. linear decay, to a set of affected `stig:Stigma` sub classes. A `stig:Law` may link to an *evolutional query* which may be used to calculate the evolution of the medium's endogenous dynamics.

[2] For example via `dct:spatial` and `dct:temporal` links.

4.2 stigFN: SPARQL Functions for Stigmergic Linked Systems

In order to facilitate the implementation of transient marker-based stigmergic Linked Systems, we supplement our domain model with the stigFN SPARQL function library. It provides the fundamental operations required for implementing the endogenous dynamics of a stigmergic medium:

1. *Decay functions.* Transient marker-based stigmergy may require certain stigmata to be subjected to dissipation processes. With `stigFN:linear_decay` and `stigFN:exponential_decay`, we provide two standard decay models.
2. *Diffusion functions.* In diffusion processes, the intensity of a stigma does not decay over time but rather spreads over a spatial dimension from the point of its deposition. With `stigFN:diffuse_1D`, the 1D diffusion equation is made available.
3. *Handling temporal and spatial values.* Decay and diffusion functions require arithmetic operations on temporal data, e.g. `xsd:duration`, `xsd:dateTime` or `xsd:time`. Due to lack of built-in support in SPARQL and XPATH, we provide `stigFN:duration_secs` and `stigFN:duration_msecs` for conversions from a `xsd:duration` value to (milli)seconds. Additionally, `stigFN:dist_manhattan` is provided as a means to find the Manhattan distance between topoi when the medium is discretised into grids.

We implemented stigFN using SPARQL user-defined functions[3] in Apache Jena[4]. https://github.com/BMBF-MOSAIK/StigLD-DemoDocumentation and source code[5] is publicly available; we intend to extend stigFN with additional decay and diffusion models as well as auxiliary functions.

5 Use Case: Make-to-Order Fulfilment

We apply the previously established concepts to a Make-to-Order (MTO) fulfilment process from the production domain. MTO is a production approach in which manufacturing starts only after a customer's order is received.

Let us consider a shop floor area that is represented by a discrete grid; in each grid cell is a shop floor location and can accommodate a single production resource. We distinguish between three types of production resources: machines, output slots assigned to individual machines and transporters.

Machines produce a product of not further specified kind in response to a confirmed order received for it from a final customer. Whenever a machine finishes production of a product, the product is placed into an output slot awaiting pickup by a transporter unit. *Output slots* have limited capacity. If any of the output slots are full, the associated machine cannot produce any new products until the output slot is emptied by the transporters. *Transporters* are initially situated in idle locations spread throughout the grid; they can move to any

[3] https://jena.apache.org/documentation/query/writing_functions.html.
[4] https://jena.apache.org/.
[5] https://github.com/BMBF-MOSAIK/StigLD-Demo.

unoccupied location within their respective Manhattan distance neighbourhood. Their task is to pick up finished products from the output slots of machines, so that production can go on without significant interruptions.

The shop floor will continuously receive new customer orders; we aim to coordinate the MTO fulfilment process such that customer orders should be assigned to machines in such a way that the overall machine work load is balanced, and make-shift times of individual products – the time from start of production to delivery of the finished product – should be minimized. More specifically, we are interested in improving the following metrics

(i) average number of steps moved by the transporters
(ii) average maximum and minimum machine loads
(iii) deviation in maximum load experienced by machines
(iv) average time between start of production of a product until pickup by a transport unit (mean time to deliver)

All material needed to set up and run the example are provided https://github.com/BMBF-MOSAIK/StigLD-Demoonline along with an http://mosaik.dfki.deinteractive demo instance[6].

5.1 Shop Floor Representation in StigLD

In our example, the `stig:Medium` represents the overall shop floor area as a 10×10 grid of `stig:Topos` instances. Neighborhood relations depend on the type of agent that is exploring the medium (see also Sect. 5.2): For transporter agents that navigate the shopfloor, each `st:Topos` links via `stig:adjacentTo` predicates to the `stig:Topos` instances in its Manhattan distance neighborhood. Order assignment agents ignore spatial information, and consider all topoi that carry a machine unit as mutually connected. Production resources are assigned to their individual `stig:Topos` instances using `stig:locatedAt` link predicates; the Transporters' idle locations – the grid cells to which they return after having finished a pickup – are given by `ex:idlePosition` link predicates.

5.2 Agent Models

We employ *marker-based stigmergy* with *transient* semio-chemical marker models to achieve the desired coordination. For this, we employ two types of agents: one type assigns open orders to available machines on the shop floor, the other controls transport units.

Order Assignment Agents: Transient Stigmergy Based on Linear Decay. For an open order, an order assignment agent $\texttt{OAA} = \texttt{PERC}(i, G = \emptyset, L = \emptyset)$ is placed on a randomly chosen topos i that is accommodating a machine; the agent performs situated perception as specified in Eq. 1 with

[6] http://mosaik.dfki.de.

$$(G'' = \texttt{ans}(q_{\text{PERC}}, G')) \equiv \left(\forall t \in G' \Rightarrow t \in G''\right)$$

$$(L'' = \texttt{sel}(q_{\text{NAV}}, G'')) \equiv (L'' = \{j \mid \underset{j}{\operatorname{argmin}} \begin{pmatrix} \texttt{<j> stig:carries [\quad stig:level ?val;} \\ \texttt{a ex:NFMarker];} \\ \texttt{\^{}(stig:locatedAt) [a ex:Machine].} \end{pmatrix} \})$$

When selecting its reaction (cf. Eq. 2)

$$\texttt{REACT}(i, G, L) = \textbf{if } i \notin L \textbf{ then } \texttt{PERC}(j \in L, \emptyset, \emptyset) \textbf{ else } \texttt{MARK}(i, G, L)$$

the agent OAA will either (i) re-situate to a topos with lower concentration of negative feedback or (ii) leave a *negative feedback marker*[7] on its current topos:

$$\texttt{MARK}(i, G, L) = \overline{req}_{\eta(i)}(\texttt{PUT}, i, \texttt{ans}(q_{\text{PUT}}, G)).res_{\eta(i)}(rc, (\emptyset, \emptyset)).\mathbf{0}$$
$$\texttt{ans}(q_{\text{PUT}}, G) \equiv \texttt{descr}(i, G) \cup \{\texttt{<i> stig:carries [a ex:NFMarker; stig:level 1.0].}\}$$

Negative feedback markers will decay linearly over time; the system's endogenous dynamics with respect to negative feedback markers is given by Eq. 3 with

$$\texttt{ans}(q_{\text{EVO},G}) \equiv \begin{pmatrix} \texttt{?i stig:carries [a ex:NFMarker; stig:level ?c; stig:decayRate ?d].} \\ \Downarrow \\ \texttt{?i stig:carries [stig:level stigFN:linear\_decay(}\Delta\texttt{t, ?d, ?c)].} \end{pmatrix}$$

Leaving a negative feedback marker *inhibits* future selection of a machine, and increases the likelihood of balancing machine workloads during the MTO process.

Transporter Agents: Transient Stigmergy Based on Diffusion. Whenever a new finished product is put into a machine's output slot, *transportation markers* (ex:TMarker) are added to the topos containing the respective slot. These markers do not decay linearly in-place, but diffuse and spread over the entire shop floor.

A transporter agent $\texttt{TA} = \texttt{PERC}(s, G = \emptyset, L = \emptyset)$ is initially situated in its idle location s; the agent performs situated perception as specified in Eq. 1 with

$$(G'' = \texttt{ans}(q_{\text{PERC}}, G')) \equiv \left(\forall t \in G' \Rightarrow t \in G''\right)$$

$$(L'' = \texttt{sel}(q_{\text{NAV}}, G'')) \equiv (L'' = \{l \mid \underset{l}{\operatorname{argmax}} \begin{pmatrix} \texttt{<l> stig:carries [stig:level ?val;} \\ \texttt{a ex:TMarker].} \end{pmatrix} \})$$

When selecting its reaction (cf. Eq. 2)

$$\texttt{REACT}(i, G, L) = \textbf{if } i \notin L \textbf{ then } \texttt{PERC}(j \in L, \emptyset, \emptyset) \textbf{ else } \texttt{PICKUP}(i, G, L)$$

$$\begin{aligned} \texttt{PICKUP}(i, G, L) = &\textbf{if } \exists p : (\texttt{<p> a \quad ex:Product; \quad stig:locatedAt \quad <i>}) \in G \\ &\textbf{then } \texttt{DEL}(p, \emptyset, \emptyset).\texttt{MOVE}(s, p).\texttt{PERC}(s, \emptyset, \emptyset) \\ &\textbf{else } \texttt{PERC}(j \in L, \emptyset, \emptyset) \end{aligned}$$

the agent TA will either (i) re-situate to a neighboring topos with higher concentration of ex:TMarker and hence climb the diffusion gradient, or (ii) attempt to pickup and move a product from its current location to its idle location.

[7] – as well as a production task into the respective machine's task queue –.

As described in Sect. 4, any `GET` request as part of a `TA` agent's situated perception (cf. Eq. 1) will trigger a diffusion update

$$\texttt{ans}(q_{\texttt{EVO},G}) \equiv \left(\begin{array}{l} \qquad\texttt{?i stig:carries [a ex:TMarker; stig:level ?c;].} \\ \qquad\qquad\qquad\qquad\qquad\Downarrow \\ \texttt{?j stig:carries [a ex:TMarker;} \\ \qquad\qquad\qquad\texttt{stig:level stigFN:diffuse1D(} \\ \qquad\qquad\qquad\texttt{?i, stigFN:dist\_manhattan(?i, ?j), ?c, } \Delta\texttt{t} \\ \qquad\qquad\qquad\texttt{)].} \end{array} \right)$$

and drive the evolution of the system's transportation markers.

Table 1. Results of simulations

	Random walk	Stigmergic coordination
Avg. number of updates	85	58
Avg. transporter steps	262	132
Mean time to deliver	112 s	67 s
Avg. max machine load	13	12
Avg. min machine load	6	8

5.3 Evaluation

We evaluated above scenario with fifty orders for products to be produced and picked up by the transporters from output slots. The shop floor contains five production machines and four transporter artifacts. For the sake of uniformity while running these simulations, all machines have output slots with a capacity of holding five finished products.

We employ the agent models as described in the previous section and benchmark against a simplified transporter agent model that only scans for finished products in its surroundings to initiate pick up, but otherwise move around randomly, i.e. not following any marker trace.

We compare the total number of updates required in each instance to complete producing fifty orders, as well as emptying them from the output slots. In addition, we compare the average number of steps moved by the transporters, the deviation in maximum load experienced by machines in each simulation and the average time that a finished product spends in an output slot before being picked up by transporters. These results can be seen in Table 1. The stigmergic coordination based shop floor simulation requires around 30% less updates in order to complete the simulation run of producing fifty orders and transporting them away from the output slots of machines. Also, it takes half as many movements by transporters compared to randomly moving transporters. Moreover, the average time it takes from a product from beginning of production to pickup by a transporter (mean time to deliver) is reduced by 40% in the stigmergy based simulation.

Average maximum and minimum machine loads are comparable in both cases, but slightly worse in the random walk simulations. Ideally, given that we have five machines and fifty orders, the average number of orders at each machine should be ten. But, since the randomly moving transporters often take longer to empty some output slots, the corresponding machines are loaded less relative to the other machines. Each update query (which includes the implicit diffusion and linear decay of stigmergic markers) takes an average of 500 milliseconds to complete.

6 Conclusions and Future Work

We propose to use a value-passing fragment of Milner's Calculus to formally specify the *generic* hypermedia-driven behaviour of Linked Data agents and the Web as their embedding environment. Based on this formalism, agents and their environment can be composed into a concurrent Linked System with declarative queries serving as extension mechanism for specifying the *domain-specific* hypermedia-driven behaviour of Linked Data agents.

Next, we took first steps into investigating stigmergic coordination principles within such Linked Systems. When considering transient marker-based stigmergy, we have identified the necessity of generating server-side effects during the handling of safe and idempotent agent-initiated resource requests. This is due to the fact that stigmergic environments may exhibit agent-less, endogenous dynamic evolution.

Based on this observation, we developed the `stigLD` domain model and the `stigFN` function library facilitating the design and declarative implementation of stigmergic principles within the agent as well as server components of a Linked System.

We demonstrate the genericity and effectiveness of our modeling approach by implementing a make-to-order (MTO) scenario from the production domain using two transient semio-chemical marker models. Our implementation displays emergence of self-organized coordination from simple agent behaviour and compares favourably against a random walk baseline strategy.

We intend to expand the `stigFN` function library with additional decay and diffusion models as well as auxiliary functions; scalability experiments and application to additional domains are subject to future work. Translating given CCS specifications of (stigmergic) Linked Systems into executable labelled-transition systems [19] is an issue for future research.

Acknowledgements. This work has been supported by the German Federal Ministry for Education and Research (BMBF) as part of the MOSAIK project (grant no. 01IS18070-C).

References

1. Alfeo, A.L., Cimino, M.G., Egidi, S., Lepri, B., Vaglini, G.: A Stigmergy-based analysis of city hotspots to discover trends and anomalies in urban transportation usage. IEEE Trans. Intell. Transp. Syst. **19**(7), 2258–2267 (2018). https://doi.org/10.1109/TITS.2018.2817558
2. Bizer, C., Heath, T., Idehen, K., Berners-Lee, T.: Linked data on the web (LDOW2008). In: Proceedings of the 17th International Conference on World Wide Web, pp. 1265–1266 (2008)
3. Charpenay, V., et al.: MOSAIK: a formal model for self-organizing manufacturing systems. IEEE Pervasive Comput. **20**, 9–18 (2020)
4. Ciortea, A., Boissier, O., Ricci, A.: Engineering world-wide multi-agent systems with hypermedia. In: Weyns, D., Mascardi, V., Ricci, A. (eds.) EMAS 2018. LNCS (LNAI), vol. 11375, pp. 285–301. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25693-7_15
5. Ciortea, A., Mayer, S., Boissier, O., Gandon, F.: Exploiting interaction affordances: on engineering autonomous systems for the web of things. In: Second W3C workshop on the Web of Things: The Open Web to Challenge IoT Fragmentation, Munich, Germany (2019)
6. Ciortea, A., Mayer, S., Gandon, F., Boissier, O., Ricci, A., Zimmermann, A.: A decade in hindsight: the missing bridge between multi-agent systems and the World Wide Web. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, vol. 3, pp. 1659–1663 (2019). https://www.alexandria.unisg.ch/256718/
7. Dipple, A., Raymond, K., Docherty, M.: General theory of Stigmergy: modelling stigma semantics. Elsevier (2014). https://doi.org/10.1016/j.cogsys.2014.02.002
8. Dipple, A.C.: Standing on the shoulders of ants: Stigmergy in the web. In: Proceedings of the 20th International Conference Companion on World Wide Web, pp. 355–360 (2011)
9. Dorigo, M., Bonabeau, E., Theraulaz, G.: Ant algorithms and Stigmergy. Future Generation Comput. Syst. **16**(8), 851–871, June 2000. https://doi.org/10.1016/S0167-739X(00)00042-X
10. Dorigo, M., Di Caro, G.: Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999, vol. 2, pp. 1470–1477. IEEE Computer Society (1999). https://doi.org/10.1109/CEC.1999.782657
11. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B Cybern. **26**(1), 29–41 (1996). https://doi.org/10.1109/3477.484436
12. Dorigo, M., Stützle, T.: Ant colony optimization: overview and recent advances. In: Gendreau, M., Potvin, J.-Y. (eds.) Handbook of Metaheuristics. ISORMS, vol. 272, pp. 311–351. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91086-4_10
13. Ferber, J., Müller, J.P.: Influences and reaction: a model of situated multiagent systems. In: 2nd International Conference on Multi-Agent Systems (ICMAS-96), pp. 72–79 (1996)
14. Fielding, R.: hypertext transfer protocol (http/1.1): semantics and content. Technical report
15. Fielding, R.: Re: draft findings on unsafe methods (whenToUseGet-7) (2002). https://lists.w3.org/Archives/Public/www-tag/2002Apr/0207.html. Accessed Apr 2021

16. Genesereth, M.R., Nilsson, N.J.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann (2012)
17. Hadeli, K., Valckenaers, P., Kollingbaum, M., Van Brussel, H.: Multi-agent coordination and control using Stigmergy. Comput. Ind. **53**(1), 75–96 (2004). https://doi.org/10.1016/S0166-3615(03)00123-4
18. Hadeli, K., et al.: Self-organising in multi-agent coordination and control using Stigmergy. In: Di Marzo Serugendo, G., Karageorgos, A., Rana, O.F., Zambonelli, F. (eds.) ESOA 2003. LNCS (LNAI), vol. 2977, pp. 105–123. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24701-2_8
19. Harth, A., Käfer, T.: Towards specification and execution of linked systems. In: GvD (2016)
20. Harth, A., Käfer, T.: Towards specification and execution of linked systems. In: CEUR Workshop Proceedings, vol. 1594, pp. 62–67 (2016)
21. Hedayatzadeh, R., Akhavan Salmassi, F., Keshtgari, M., Akbari, R., Ziarati, K.: Termite colony optimization: a novel approach for optimizing continuous problems. In: 2010 18th Iranian Conference on Electrical Engineering, pp. 553–558 (2010). https://doi.org/10.1109/IRANIANCEE.2010.5507009
22. Heylighen, F.: Stigmergy as a generic mechanism for coordination: definition, varieties and aspects. Cognition, pp. 1–23 (2011)
23. Heylighen, F.: Stigmergy as a universal coordination mechanism I: definition and components. Cogn. Syst. Res. **38**, 4–13 (2016). https://doi.org/10.1016/j.cogsys.2015.12.002
24. Heylighen, F.: Stigmergy as a universal coordination mechanism II: varieties and evolution. Cogn. Syst. Res. **38**, 50–59 (2016). https://doi.org/10.1016/j.cogsys.2015.12.007
25. Heylighen, F., Vidal, C.: Getting things done: the science behind stress-free productivity. Long Range Plan. **41**(6), 585–605 (2008)
26. Hunt, E.R., Jones, S., Hauert, S.: Testing the limits of pheromone Stigmergy in high-density robot swarms. Roy. Soc. Open Sci. **6**(11) (2019). https://doi.org/10.1098/rsos.190225
27. Jevtić, A., Gutierrez, Á., Andina, D., Jamshidi, M.: Distributed bees algorithm for task allocation in swarm of robots. IEEE Syst. J. **6**(2), 296–304 (2012). https://doi.org/10.1109/JSYST.2011.2167820
28. Jochum, B., Nürnberg, L., Aßfalg, N., Käfer, T.: Data-driven workflows for specifying and executing agents in an environment of reasoning and RESTful systems. In: Di Francescomarino, C., Dijkman, R., Zdun, U. (eds.) BPM 2019. LNBIP, vol. 362, pp. 93–105. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-37453-2_9
29. Kanamori, R., Takahashi, J., Ito, T.: Evaluation of traffic management strategies with anticipatory stigmergy. J. Inf. Process. **22**(2), 228–234 (2014). https://doi.org/10.2197/ipsjjip.22.228
30. Krieger, M.J., Billeter, J.B., Keller, L.: Ant-like task allocation and recruitment in cooperative robots. Nature **406**(6799), 992–995 (2000). https://doi.org/10.1038/35023164
31. Milner, R. (ed.): A Calculus of Communicating Systems. LNCS, vol. 92. Springer, Heidelberg (1980). https://doi.org/10.1007/3-540-10235-3
32. Milner, R.: Communication and concurrency. PHI Series in Computer Science. Prentice Hall (1989)
33. Privat, G.: Phenotropic and stigmergic webs: the new reach of networks. Univ. Access Inf. Soc. **11**(3), 323–335 (2012). https://doi.org/10.1007/s10209-011-0240-1

34. Smith, G.J., Gero, J.S.: What does an artificial design agent mean by being 'situated'? Des. Stud. **26**, 535–561 (2005). https://doi.org/10.1016/j.destud.2005.01.001
35. Spieldenner., T., Chelli., M.: Linked data as stigmergic medium for decentralized coordination. In: Proceedings of the 16th International Conference on Software Technologies - ICSOFT, pp. 347–357. INSTICC. SciTePress (2021). https://doi.org/10.5220/0010518003470357
36. Theraulaz, G., Bonabeau, E.: A brief history of stigmergy. Artif. Life **5**(2), 97–116 (1999)
37. Tummolini, L., Castelfranchi, C.: Trace signals: the meanings of stigmergy. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 141–156. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71103-2_8
38. Valckenaers, P., Kollingbaum, M., Van Brussel, H., et al.: Multi-agent coordination and control using stigmergy. Comput. Ind. **53**(1), 75–96 (2004)
39. Valckenaers, P., Van Brussel, H., Kollingbaum, M., Bochmann, O.: Multi-agent coordination and control using stigmergy applied to manufacturing control. In: Luck, M., Mařík, V., Štěpánková, O., Trappl, R. (eds.) ACAI 2001. LNCS (LNAI), vol. 2086, pp. 317–334. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47745-4_15
40. Dyke Parunak, H.: A survey of environments and mechanisms for human-human stigmergy. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2005. LNCS (LNAI), vol. 3830, pp. 163–186. Springer, Heidelberg (2006). https://doi.org/10.1007/11678809_10
41. Weyns, D., Agentwise, T.H.: A formal model for situated multi-agent systems. Technical report (2004)
42. Weyns, D., Holvoet, T.: Model for simultaneous actions in situated multi-agent systems. In: Schillo, M., Klusch, M., Müller, J., Tianfield, H. (eds.) MATES 2003. LNCS (LNAI), vol. 2831, pp. 105–118. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39869-1_10
43. Weyns, D., Omicini, A., Odell, J., Weyns, D., Omicini, A., Odell, J.: Environment as a first class abstraction in multiagent systems model of the environment. Auton. Agent Multi-Agent Syst. **14**(1), 5–30 (2007). https://doi.org/10.1007/s10458-006-0012-0