mathematics

MDPI

Article

stigLD: Stigmergic Coordination in Linked Systems †

René Schubotz [1,*,‡], Torsten Spieldenner [1,2,*,‡] and Melvin Chelli [1,*,‡]

[1] German Research Center for Artificial Intelligence, Saarland Informatics Campus D3 2, 66123 Saarbrücken, Germany
[2] Saarbrücken Graduate School of Computer Science, Campus E1 3, 66123 Saarbrücken, Germany
* Correspondence: rene.schubotz@dfki.de (R.S.); torsten.spieldenner@dfki.de (T.S.); melvin.chelli@dfki.de (M.C.)

† The original version of this paper was presented at the 16th International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA 2021), December 2021. This paper was recommended for publication in revised form by the BIC-TA 2021 conference committees.

‡ These authors contributed equally to this work.

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 of modelling approach in a Make-to-Order fulfilment scenario involving transient stigmergy and negative feedback as well as by solving a problem instance from the (time constrained) Trucks World domain as presented in the fifth International Planning Competition.

Keywords: Linked Data; Semantic Web; multi-agent systems; stigmergy; nature inspired algorithm; RDF; SPARQL; biologically inspired computing

MSC: 68Q07

check for updates

Citation: Schubotz, R.; Spieldenner, T.; Chelli, M. stigLD: Stigmergic Coordination in Linked Systems. *Mathematics* **2022**, *10*, 1041. https://doi.org/10.3390/math10071041

Academic Editors: Linqiang Pan, Zhihua Cui, Harish Garg, Thomas Hanne and Gai-Ge Wang

Received: 31 January 2022
Accepted: 22 March 2022
Published: 24 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Hypermedia multi-agent systems [1,2], sometimes also referred to as Linked Systems [3], 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 closely aligned with REST and Linked Data best practices [4], 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 [5,6] provides an indirect and mediated feedback mechanism between agents, and enables complex, 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 [7] given that "it is its mediating function that underlies the power of stigmergy" [5]. 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. In this context, we are in particular interested in how to employ Linked Data media to provide environments with immanent dynamics, a core feature for stigmergic systems that display, for example, transient or diffusion-based marker behavior. By this, the interaction between agents and environments should be described unambiguously, generically, and implementation-agnostically to allow application of the found concepts in a variety of system implementations. Towards this end, this paper makes the following contributions:

- A definition of Linked Data servers, Linked Data agents, and the interaction between those using Milner's Calculus of Communicating Systems (CCS).
- A definition of dynamic Linked Data Media servers to account for immanent dynamics of stigmergic environments.
- An RDF domain model for dynamic stigmergic environments, along with a set of SPARQL functions to implement transient and diffusion-based marker behavior.
- Examples of application of the presented concepts from the domains of shop-floor logistics, and optimization.

The remainder of this chapter is structured as follows: We briefly present core concepts and variations of stigmergic systems and summarise existing literature relevant to our work in Section 2. Next in Section 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 a coordination mechanism between Linked Data agents in Section 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, in Section 6 we solve instances of the *(time constrained) Trucks World domain* as presented in the fifth International Planning Competition (IPC-5) [8] and find our approach to be on-par with the leading planners in this IPC-5 domain. We conclude and point out future work in Section 7.

2. Varieties of Stigmergy and Related Work

In collective stigmergic systems, groups of *agents* perform work by executing *actions* within their environment [5]. 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 their conditions are 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* [9]. 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 a 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 Figure 1).

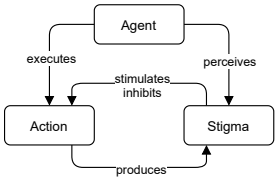

Figure 1. Stigmergic feedback loop.

In essence, stigmata work as an indirect communication mechanism between agents [10], potentially leading to coordination between agents, and, ideally, a self-organising behaviour of the entire system [5,6]. Based on these core concepts, i.e., *action*, *medium* and *stigma*, stigmergic systems can be further classified as follows [5]. 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 the 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 [11]. 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 optimisation [5,6,12].

Stigmergy is recognised as a suitable underlying principle for multi-agent systems [10,13,14] and is applied in a variety of practical domains, e.g., digital manufacturing [15], robotics [16,17] or public transport [18,19].

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 [20–22]. 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 [23].

Web technologies have been found to be a suitable basis for implementation of multi agent systems [24–26]. Meanwhile, it came to attention that stigmergic principles are the underlying concept of many applications in the World Wide Web [27] including coordination in Web-based IoT systems [28].

Self-organising multi agent systems and agent systems that rely on stigmergy as coordination mechanism have been exhaustively reviewed in [29]. This review concludes that a common understanding of such systems is widely lacking, and suggests a generic domain model to describe self-organising systems. From the reviewed literature, we share this observation, and additionally identify a general lack of understanding of the importance of a properly defined digital medium in multi-agent systems, in particular in systems that implement stigmergic principles. The interaction between agents and environment is often described only vaguely, and is generally under-specified.

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) [30,31]. 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* [3] or a hypermedia MAS [1].

3.1. Theoretical Setting: CCS with Value-Passing

The Calculus of Communicating Systems (CCS) is a process calculus that models indivisible communication between two participants as actions. The formal language

includes primitives for describing parallel composition, choice between actions and scope restriction. CCS is useful for evaluating the qualitative correctness of system properties such as deadlock or live-lock. For an in-depth exposition we refer the interested reader to [30,31] and continue with a self-contained introduction to the core concepts necessary for our means.

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 abstract syntax given in Figure 2. 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; $P_{1 \leq i \leq 2} \in \mathcal{P}$ are process expressions; and e is a Boolean expression.

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

Figure 2. Abstract syntax of all CCS process expressions.

A *process definition* is an equation system of the form $(K_{1 \leq i \leq k} = P_{1 \leq i \leq k})$ where $P_{1 \leq i \leq k} \subset \mathcal{P}$ is a set of process expression with process identifiers from $K_{1 \leq i \leq k} \subset \mathcal{K}$.

Each process definition determines an Act-labelled transition system whose transitions can be inferred from the Structural Operational Semantics rules given in Figure 3 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.

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

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

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

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

Figure 3. Structural Operational Semantics rules.

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 $\texttt{ans} : Q \times 2^{\mathcal{T}} \to 2^{\mathcal{T}}$, $\texttt{ask} : Q \times 2^{\mathcal{T}} \to \mathbb{B}$, $\texttt{sel} : Q \times 2^{\mathcal{T}} \to 2^{\mathbf{I}}$ and $\texttt{descr} : \mathbf{I} \times 2^{\mathcal{T}} \to 2^{\mathcal{T}}$.

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

3.2.1. Linked Data Servers

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

$$\texttt{SERVER}_k(G) = req_k(op, i, G').\texttt{PROC}_k(op, i, G', G) \tag{1}$$

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

$$\texttt{PROC}_k(\texttt{GET}, i, G', G) = \texttt{RESP}_k(\texttt{OK}, (\emptyset, \texttt{descr}(i, G)), G) \; + \; \texttt{RESP}_k(\texttt{ERR}, (\emptyset, \emptyset), G) \tag{2}$$

$$\texttt{PROC}_k(\texttt{PUT}, i, G', G) = \texttt{RESP}_k(\texttt{OK}, (\emptyset, \emptyset), (G \setminus \texttt{descr}(i, G)) \cup G') \; + \; \texttt{RESP}_k(\texttt{ERR}, (\emptyset, \emptyset), G) \tag{3}$$

$$\texttt{PROC}_k(\texttt{POST}, i, G', G) = \texttt{RESP}_k(\texttt{OK}, (\{i'\}, \emptyset), G \cup G') \; + \; \texttt{RESP}_k(\texttt{ERR}, (\emptyset, \emptyset), G) \tag{4}$$

$$\texttt{PROC}_k(\texttt{DEL}, i, G', G) = \texttt{RESP}_k(\texttt{OK}, (\{i\}, \emptyset), G \setminus \texttt{descr}(i, G)) \; + \; \texttt{RESP}_k(\texttt{ERR}, (\emptyset, \emptyset), G) \tag{5}$$

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

$$\texttt{RESP}_k(rc, rval, G) = \overline{res}_k(rc, rval).\texttt{SERVER}_k(G) \tag{6}$$

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

3.2.2. Tropistic Linked Data Agents

We specify a *tropistic* ([32], Section 13.1) Linked Data agent $\texttt{AGENT}_k$ as an active component

$$\texttt{AGENT}_k = \texttt{PERC}_k(i \in R, G = \emptyset, L = \{i\}) \tag{7}$$

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 $\texttt{AGENT}_k$ puts emphasis on a direct response to its perceptions and favours to employ *situated perceptions* [33] of the environment as the basis for deciding which action to perform next. We model situated perception in CCS-style as

$$\begin{aligned} \texttt{PERC}_k(i, G, L) = & \overline{req}_{\eta(j)}(\texttt{GET}, j, \emptyset).res_{\eta(j)}(rc, (L', G')). \\ & \Big(\texttt{PERC}_k(i, G'', L'') + \texttt{REACT}_k(i, G'', L'') \Big) \end{aligned} \tag{8}$$

where $\texttt{AGENT}_k$ - while being situated at i - will at first issue a $\texttt{GET}$ 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

$$G'' = G \cup \texttt{ans}(q_{\texttt{PERC}_k}, G') \tag{9}$$

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

$$L'' = L \cup L' \cup \texttt{sel}(q_{\texttt{NAV}_k}, G'')) \tag{10}$$

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

$$\texttt{REACT}_k(i,G,L) = \texttt{PERC}_k(j \in L, \emptyset, \{j\}) + \sum_{m \in \texttt{OPS} \setminus \{\texttt{GET}\}} \left(\textbf{if } \texttt{ask}(\widehat{q}_{\texttt{m}_k}, G, L) \textbf{ then } \texttt{m}_k(i,G,L) \textbf{ else } \texttt{REACT}_k(i,G,L) \right) \tag{11}$$

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

$$\texttt{PUT}_k(i,G,L) = \overline{req}_{\eta(i)}(\texttt{PUT}, i, \texttt{ans}(q_{\texttt{PUT}_k}, G)).res_{\eta(i)}(rc, (\emptyset, \emptyset)).\texttt{REACT}_k(i,G,L) \tag{12}$$

$$\texttt{POST}_k(i,G,L) = \overline{req}_{\eta(i)}(\texttt{POST}, i, \texttt{ans}(q_{\texttt{POST}_k}, G)).res_{\eta(i)}(rc, (L', \emptyset)).\texttt{REACT}_k(i,G,L \cup L') \tag{13}$$

$$\texttt{DEL}_k(i,G,L) = \overline{req}_{\eta(i)}(\texttt{DEL}, i, \emptyset).res_{\eta(i)}(rc, (L', \emptyset)).\texttt{REACT}_k(j \in L \setminus L', G, L \setminus L') \tag{14}$$

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 [1] or a *Linked System* [3].

3.2.3. Linked Systems

A *Linked System* [3] is the parallel composition

$$\texttt{LINKED-SYSTEM} \quad = \quad (\texttt{AGENTS} \parallel \texttt{ENVIRONMENT}) \tag{15}$$

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 Section 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 behaviours as described by the CCS fragment are generic and independent of the scenarios in which they are applied. Domain- or application-specific behaviours 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 [34–37] and termite colony optimisation methods [38], 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 ([5], 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 ([39], cf. Figure 4), and more importantly, they are *not* subjected to agent-driven processes. 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*.

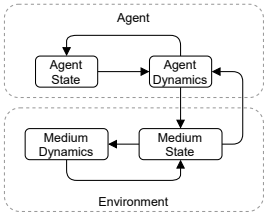

Figure 4. Stigmergic system components.

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

$$\texttt{STIGMERGIC-LINKED-SYSTEM} \quad = \quad (\texttt{AGENTS} \parallel (\texttt{MEDIUM} \parallel \texttt{ENVIRONMENT})) \tag{16}$$

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 (We emphasise that this conception is not in violation with HTTP semantics ([40], Sections 4.2.1 and 4.2.2; [41])).

$$\texttt{MEDIUM}_k(G) = req(op, i, G').\texttt{PROC}_k(op, i, G', G)) \tag{17}$$

$$\texttt{RESP}_k(rc, rval, G) = \overline{res}(rc, rval).\texttt{MEDIUM}_k(G) \tag{18}$$

$$\texttt{PROC}_k(\texttt{GET}, i, G', G) = \texttt{EVOLVE}_k(i, G) \tag{19}$$

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

$$\texttt{EVOLVE}_k(i, G) = \texttt{RESP}(\texttt{OK}, (\emptyset, \texttt{descr}(i, G')), G'') \; + \; \texttt{RESP}_k(\texttt{ERR}, (\emptyset, \emptyset), G) \tag{20}$$

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.

4.1. `stigLD`: A Domain Model for Stigmergic Linked Systems

Our domain model (cf. Figure 5) defines four basic concepts: `stig:Medium`, `stig:Law`, `stig:Topos` and `stig:Stigma`.

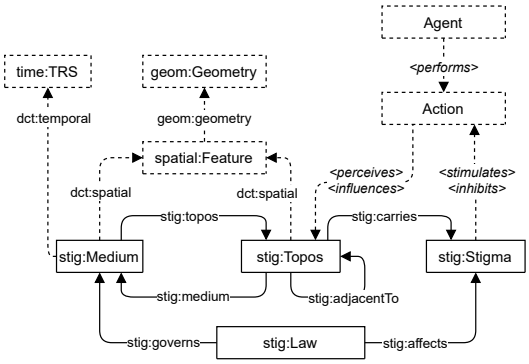

Figure 5. `stigLD` domain model.

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" [5], 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` instances governing its endogenous dynamics.

A `stig:Medium` may optionally detail on its spatio-temporal characteristics (for example, via `dct:spatial` and `dct:temporal` links.), 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` instances (cf. Figure 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.

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. We provide two standard decay models with `stigFN:linear_decay` and `stigFN:exponential_decay`.
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 of `xsd:duration` datatype values. 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 (https://jena.apache.org/documentation/query/writing_functions.html, accessed on 21 March 2022) in Apache Jena (https://jena.apache.org/, accessed on 21 March 2022). Documentation and source code (https://github.com/BMBF-MOSAIK/StigLD-Demo, accessed on 21 March 2022) 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 an unspecified 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 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 online (https://github.com/BMBF-MOSAIK/StigLD-Demo, accessed on 21 March 2022) along with an interactive demo instance (https://mosaik.dfki.de, accessed on 21 March 2022).

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 Section 5.2): For transporter agents that navigate the shopfloor, each `stig: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.

5.2.1. 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 Equation (8) with

$$(G'' = \texttt{ans}(q_{\texttt{PERC}}, G')) \equiv (\forall t \in G' \Rightarrow t \in G'') \tag{21}$$

$$(L'' = \texttt{sel}(q_{\texttt{NAV}}, G'')) \equiv (L'' = \{j \,|\, \underset{j}{\text{argmin}} \begin{pmatrix} \texttt{<j>} & \texttt{stig:carries [} & \texttt{stig:level ?val;} \\ & & \texttt{a ex:NFMarker];} \\ & \texttt{\^{}(stig:locatedAt)} & \texttt{[a ex:Machine].} \end{pmatrix}\}) \tag{22}$$

When selecting its reaction (cf. Equation (11))

$$\texttt{REACT}(i, G, L) = \textbf{if } i \notin L \textbf{ then } \texttt{PERC}(j \in L, \emptyset, \emptyset) \textbf{ else } \texttt{MARK}(i, G, L) \tag{23}$$

the agent `OAA` will either (i) re-situate to a topos with lower concentration of negative feedback or (ii) leave a negative feedback marker(as well as a production task into the respective machine's task queue) on its current topos:

$$\texttt{MARK}(i, G, L) = \overline{req}_{\eta(i)}(\texttt{PUT}, i, \texttt{ans}(q_{\texttt{PUT}}, G)).res_{\eta(i)}(rc, (\emptyset, \emptyset)).\mathbf{0} \tag{24}$$

$$\texttt{ans}(q_{\texttt{PUT}}, G) \equiv \texttt{descr}(i, G) \cup \{\texttt{<i> stig:carries [a ex:NFMarker; stig:level 1.0].}\} \tag{25}$$

Negative feedback markers will decay linearly over time; the system's endogenous dynamics with respect to negative feedback markers is given by Equation (20) with

$$q_{\texttt{EVO}} \equiv \begin{pmatrix} \texttt{?i stig:carries [a ex:NFMarker; stig:level ?c; stig:decayRate ?d].} \\ \Downarrow \\ \texttt{?i stig:carries [stig:level stigFN:linear\_decay(}\Delta\texttt{t, ?d, ?c)].} \end{pmatrix} \tag{26}$$

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

5.2.2. 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 Equation (8) with

$$(G'' = \texttt{ans}(q_{\texttt{PERC}}, G')) \equiv \left(\forall t \in G' \Rightarrow t \in G''\right) \tag{27}$$

$$(L'' = \texttt{sel}(q_{\texttt{NAV}}, G'')) \equiv (L'' = \{?l \,|\, \underset{?c}{\text{argmax}}\left(\begin{array}{ll}\texttt{?l stig:carries [} & \texttt{stig:level ?c;} \\ & \texttt{a ex:TMarker].}\end{array}\right)\}) \tag{28}$$

When selecting its reaction (cf. Equation (11))

$$\texttt{REACT}(i, G, L) = \textbf{if } i \notin L \textbf{ then } \texttt{PERC}(j \in L, \emptyset, \emptyset) \textbf{ else } \texttt{PICKUP}(i, G, L) \tag{29}$$

$$\begin{aligned}\texttt{PICKUP}(i, G, L) = \textbf{if } &\exists p : (\texttt{<p> a ex:Product; stig:locatedAt <i>}) \in G \\ &\textbf{then } \texttt{DEL}(p, \emptyset, \emptyset).\texttt{MOVE}(s, p).\texttt{PERC}(s, \emptyset, \emptyset) \\ &\textbf{else } \texttt{PERC}(j \in L, \emptyset, \emptyset)\end{aligned} \tag{30}$$

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.

As described in Section 4, any GET request as part of a TA agent's situated perception (cf. Equation (8)) will trigger a diffusion update

$$q_{\texttt{EVO}} \equiv \left(\begin{array}{l}\texttt{?i stig:carries [a ex:TMarker; stig:level ?c;].} \\ \qquad\qquad\qquad\qquad\qquad\Downarrow \\ \texttt{?j stig:carries [a ex:TMarker;} \\ \qquad\qquad\qquad\quad \texttt{stig:level stigFN:diffuse1D(} \\ \qquad\qquad\qquad\quad \texttt{?i, stigFN:dist\_manhattan(?i, ?j), ?c, } \Delta\texttt{t} \\ \qquad\qquad\qquad\quad \texttt{)].}\end{array}\right) \tag{31}$$

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

5.3. Evaluation

We evaluated the 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 artefacts. 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.

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

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 ms to complete.

6. Use Case: IPC Trucks

In this section, we demonstrate how to achieve *self-optimising* behavior by application of dynamic stigmergic markers, and the `stigLD` domain model. For this, we implement a stigmergic agent system to solve the *(time constrained) trucks problem* as presented in the fifth International Planning Competition (IPC-5) [8]. The International Planning Competition is an annual contest to evaluate various planners on pre-defined sets of problems. Evaluating our stigmergic approach against problems from these competitions thus allows us to compare the performance of our system against a set of established and publicly available benchmarks.

The chosen *trucks world* problem consists of a number of mutually connected *locations*. Initially, any of the locations may carry a number of *packages*. Locations moreover may serve as *destinations* for packages, as determined by a set of *orders*. The objective is for a truck to pick up packages and deliver them to their final location in minimal time.

Travelling from one location to another takes the truck a certain amount of time, depending on the distance between two locations, as set by the specific scenario. The truck's loading capacity is limited, and divided into a set of *areas* that can carry one package each. The truck's loading bay needs to be loaded and unloaded in a last in, first out manner, i.e., packages closer to the end of the truck block the unloading of packages farther inside the truck. Loading packages to, unloading from, and finally delivering packages at the destination, each take an action with a fixed time span considerably shorter than the driving times between locations.

Figure 6 shows a simple instance of the trucks world problem that consists of three locations and two packages, and the respective optimal plan for the displayed problem.

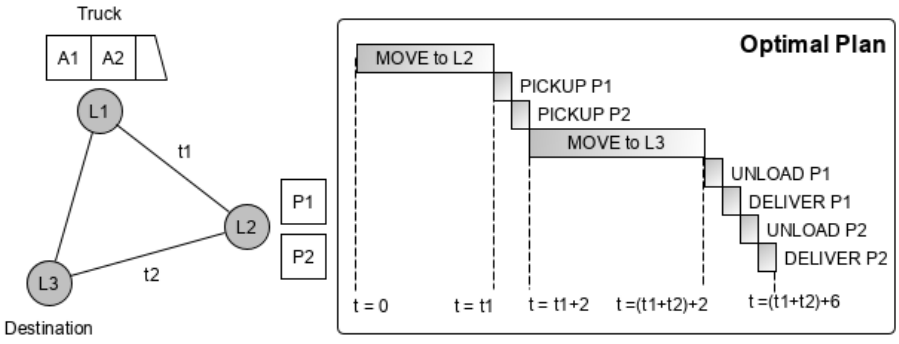

Figure 6. Schematic picture of the trucks world problem, and a Gantt diagram of a respective plan satisfying the problem constraints.

The IPC-5 trucks world problem comes with a series of variants each of which introduces additional constrains to be considered during planning. We evaluated our system against the *time constrained* trucks world problem. In this variant, each delivery order comes with a deadline by which a package has to be delivered. If the truck fails to deliver any package within its specified deadline, the problem is considered as not solved.

This IPC-5 time constrained trucks world problem is particularly challenging to solve for self-coordinating systems, as the presented tropistic agents (cf. Section 3.2.2) do not maintain long-term internal models of the environment nor have capabilities to look ahead in time, trace back sequences of actions to find the most optimal out of a number of explored solutions, or explore several potential solutions in parallel.

The time constrained trucks world problem moreover allows us to judge the efficacy of our system w.r.t to self-coordination by solely observing if it manages to keep the given deadlines or not, without necessarily aiming for the shortest, (self-)optimised time of completion.

All material needed to replicate the presented experiments and reported results is available online (https://github.com/dfki-asr/stigld-trucks-world, accessed on 21 March 2022).

6.1. Trucks World Representation in StigLD

We represent the *locations* of a trucks world problem instance as both a `stig:Topos` instance (We use `stig:` as the predefined namespace prefix for the `stigLD` vocabulary.) and `trucks:Location` (We introduce `trucks:` as a namespace prefix for a IPC-5 trucks world domain vocabulary.) instance. Locations are mutually linked by the predicate `stig:adjacentTo`, and moreover specify the distance to other locations by triples that are linked via the predicate `trucks:driveTime` (see also Listing 1). *Trucks* are linked to their current location by the `stig:locatedAt` predicate.

Listing 1: A location in the trucks world domain, represented in the `stigLD` domain model.

```
:loc1 a stig:Topos, trucks:Location  ;
  stig:adjacentTo   :loc2, :loc3;
  trucks:driveTime   [ trucks:destination :loc2;
                       rdf:value  "406.3"^^xsd:double];
  trucks:driveTime   [ trucks:destination :loc3;
                       rdf:value  "73.1"^^xsd:double].
```

The truck's loading bay areas are represented as both `trucks:Area` and `stig:Topos` instances, and denote their position in the truck via the predicate `trucks:position`, linking to an integer value. Areas are sorted from the front to the back of the truck by increasing `trucks:position` values, as also shown in Figure 6. Packages are represented as both `trucks:Package` and `stig:Topos` instances as they may carry stigmergic markers as result of `MEDIUM`'s evolution (cf. Section 6.2). Packages are assigned to their current location via the predicate `stig:adjacentTo`, i.e., the triple (<p> `stig:adjacentTo` <l>) denotes

that a package p is currently located at a location l. Delivery orders are represented as `trucks:Goal` instances that specify the package to be delivered via a `trucks:payload` predicate, the target location using a `trucks:destination` predicate, and potential deadlines through the `trucks:deadline` predicates.

A given trucks world problem instance is translated into an RDF graph G representation and maintained by a `MEDIUM` server. The `MEDIUM` offers a constrained set of operations to access and manipulate artefacts of the trucks world problem instance, and will generate server-side side-effects that we detail in the following.

6.2. Marker Model

We employ markers to signify packages requiring urgent pickup or delivery, i.e., for which time runs short to meet the deadline. In essence, each package marker is assigned a concentration level from the unit interval as soon as the remaining time buffer for successful deliver falls below the confidently estimated delivery time.

The concentration $c_i(t)$ of an individual package marker i is a function of the current simulation time t

$$c_i(t) = \begin{cases} \frac{1}{1-r_i} & \text{iff } r_i < (\delta_p^i + \delta_d^i) * \gamma \\ 0 & \text{otherwise} \end{cases} \tag{32}$$

where the term δ_p^i denotes the truck's drive time from its current location to the pickup location of package i, the term δ_d^i denotes the truck's drive time from the respective pickup to the destination of package i, γ is a configurable *confidence parameter* (In our evaluations we have chosen $\gamma = 1.5$. The value was determined empirically.), d_i is the package's deadline, and $r_i = min(0, d_i - t - (\delta_p^i + \delta_d^i))$ is the remaining time buffer for successful package deliver.

The RDF graph G encoding of the problem instance is maintained by the stigmergic system's `MEDIUM` component. As described in Section 4, any `GET` request as part of a `TRUCK` agent's situated perception (cf. Equation (8)) will trigger the following server-side updates

$$q_{\texttt{EVO}} \equiv \begin{pmatrix} \texttt{?i a trucks:Package, stig:Topos .} \\ \Downarrow \\ \texttt{?i stig:carries [a stig:Stigma; stig:level } c_i(t) \texttt{].} \end{pmatrix} \tag{33}$$

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

6.3. Truck Agent Model

A truck agent $\texttt{TRUCK} = \texttt{PERC}(u, G = \emptyset, L = \emptyset)$ is initially situated in a location i, specified by the scenario description. Situatedness of the agent on a resource i is expressed by a triple (`:truck stig:locatedAt <i>`). The agent performs situated perception as specified in Equation (8) with

$$(G'' = \texttt{ans}(q_{\texttt{PERC}}, G')) \equiv (\forall t \in G' \Rightarrow t \in G'') \tag{34}$$

Let i be the resource the agent is currently situated on, and `:trucks` denote the resource representing the `TRUCKS` agent. When selecting its reaction (cf. Equation (11)), the agent may perform a `PICKUP`, if situated on a resource representing a package, or `DROP` and `DELIVER` a package from its bay, if situated on a location that receives a package, with:

$$\begin{aligned} \texttt{REACT}(i, G, L) = & \textbf{if } i \notin L \textbf{ then } \texttt{PERC}(j \in L, \emptyset, \emptyset) \\ & \textbf{elseif } (< \texttt{i} > \texttt{a trucks:Location}) \in G \textbf{ then } \texttt{DROP}(i, G).\texttt{DELIVER}(i, G) \\ & \textbf{elseif } (< \texttt{i} > \texttt{a trucks:Package}) \in G \textbf{ then } \texttt{PICKUP}(i, G) \end{aligned} \tag{35}$$

$$\texttt{PICKUP(i,G)} \;=\; \texttt{MOVE(:truck, l) ; (<i> stig:adjacentTo <l>)} \in G \tag{36}$$

$$\texttt{MOVE(i,a) ; a} \;=\; \underset{?p}{\texttt{argmax<a>}} \begin{pmatrix} \texttt{<a>} & \texttt{a trucks:Area;} \\ & \texttt{trucks:position ?p ;} \\ & \texttt{trucks:status trucks:empty .} \end{pmatrix} \tag{37}$$

$$\begin{aligned} \texttt{DELIVER(i,G)} = \textbf{if } & \begin{pmatrix} \texttt{<p>} & \texttt{a trucks:Package ;} \\ & \texttt{stig:adjacentTo <i> .} \end{pmatrix}, \begin{pmatrix} \texttt{<g>} & \texttt{a trucks:Goal ;} \\ & \texttt{trucks:payload <p>;} \\ & \texttt{trucks:destination <i> .} \end{pmatrix} \in \texttt{G} \\ \textbf{then } & \texttt{DEL(i,}\emptyset,\emptyset\texttt{)} \end{aligned} \tag{38}$$

$$\begin{aligned} \texttt{DROP(i,G)} = \textbf{if } & \begin{pmatrix} \texttt{<p>} & \texttt{a trucks:Package ;} \\ & \texttt{stig:adjacentTo <a> .} \\ \texttt{<a>} & \texttt{a trucks:Area .} \end{pmatrix}, \begin{pmatrix} \texttt{<g>} & \texttt{a trucks:Goal ;} \\ & \texttt{trucks:payload <p>;} \\ & \texttt{trucks:destination <i> .} \end{pmatrix} \in \texttt{G} \\ \textbf{or if } & \begin{pmatrix} \texttt{<p>} & \texttt{a trucks:Package ;} \\ & \texttt{stig:adjacentTo <i> ;} \\ & \texttt{stig:carries [a stig:Stigma ;} \\ & \qquad \texttt{stig:level ?c}_p \texttt{] .} \end{pmatrix} \in G \text{ , } ?c_p > 0, \\ \textbf{and } & \begin{pmatrix} \texttt{<q>} & \texttt{a trucks:Package ;} \\ & \texttt{stig:adjacentTo [a trucks:Area] ;} \\ & \texttt{stig:carries [a stig:Stigma ;} \\ & \qquad \texttt{stig:level ?c}_q \texttt{] .} \end{pmatrix} \in G \text{ , } ?c_q = 0, \\ \textbf{and } & \left| \begin{pmatrix} \texttt{<a>} & \texttt{a trucks:Area ;} \\ & \texttt{trucks:status :empty .} \end{pmatrix} \in G \right| < |\texttt{<p>}|, \\ \textbf{then } & \texttt{MOVE(k,i) ; k} = \underset{?pos}{\texttt{argmin<k>}} \begin{pmatrix} \texttt{<k>} & \texttt{a truck:Package ;} \\ & \texttt{stig:adjacentTo <a> .} \\ \texttt{<a>} & \texttt{a trucks:Area ;} \\ & \texttt{trucks:position ?pos} \end{pmatrix} \end{aligned} \tag{39}$$

During PICKUP, the truck agent will move the package, on which it is currently situated, from its current location to the furthest back free loading area in the bay, i.e., insert a triple (<i> stig:adjacentTo <a>), and subsequently relocate the truck to the location from which it picked up the package.

Conversely, DROP will move the front-most package from the loading bay to the location on which the truck is currently located. The truck will also DROP the front-most package as long as there are more urgent packages located on i, than the truck can currently accommodate, but the space in the truck is occupied with non urgent packages. If the location i on which the truck is situated is destination to any of the packages located on i, they are DELIVERed, i.e., removed from the scenario, and the respective delivery goal is marked as completed. Note that whenever the truck reaches a location to which a package needs to be delivered, the truck prioritises unloading and delivering over any pickup action.

PICKUP, DROP and DELIVER increase the simulation time clock by the duration for these action as defined by the scenario. (In the evaluated use cases, PICKUP, DROP and DELIVER take 1 time step each.)

If none of the conditions above apply, the truck will update its linkset and relocate to another resource (cf. Equation (10)):

Let $(L'' = \texttt{sel}(q_{\texttt{NAV}}, G'')) \equiv L'' = n(\circ)$ denote the evaluation of the navigational query, then as a result of perceiving i, and neighbouring locations j, the agent may further decide to navigate to another location as follows:

1. If (<p> stig:adjacentTo <i>) $\in$ G , i.e, if there is any package on the trucks current location, then:

$$n(\circ) = \underset{?dist}{\text{argmax}}\{\underset{?c}{\text{argmax}}_{|a|}\{p \mid \begin{pmatrix} \texttt{<i>} & \texttt{a trucks:Location ;} \\ & \texttt{trucks:driveTime [trucks:destination ?q ;} \\ & \texttt{rdf:value ?dist] .} \\ \texttt{<p>} & \texttt{stig:adjacentTo <i> ;} \\ & \texttt{stig:carries [a stig:Stigma ;} \\ & \texttt{stig:level ?c] .} \\ \texttt{<a>} & \texttt{a trucks:Area ;} \\ & \texttt{trucks:status :empty .} \\ \texttt{<g>} & \texttt{a trucks:Goal ;} \\ & \texttt{trucks:destination <d> ;} \\ & \texttt{trucks:payload <p> .} \end{pmatrix} \in G\}\} \quad (40)$$

Consequently, out of the $|a|$ *most urgent* packages, with $|a|$ being the number of free loading bay areas in the truck, the truck moves to the package with the *farthest distance to its destination*, leading to loading urgent parcels with nearby delivery destinations last.

2. If above condition is not satisfied, i.e., there are no packages on i or the loading bay is full, then if for any package p with (<p> stig:adjacentTo [a trucks:Area]) $\in G$ there is (<p> stig:carries a stig:Stigma ; stig:level ?c]) $\in$ G with $c > 0$, i.e., any package within the truck's loading bay has exceeded it's delivery confidence value:

$$n(\circ) = \underset{?c}{\text{argmax}}\texttt{l}\{l \mid \begin{pmatrix} \texttt{<p>} & \texttt{a trucks:Package ;} \\ & \texttt{stig:carries [a stig:Stigma ;} \\ & \texttt{stig:level ?c] .} \\ \texttt{<g>} & \texttt{a stig:Goal ;} \\ & \texttt{trucks:destination <l> ;} \\ & \texttt{trucks:payload <p> .} \end{pmatrix} \in G, c > 0\}\} \quad (41)$$

Consequently, the truck moves to the next location l to which a package needs to be delivered most urgently.

3. If none of the packages needs to be delivered urgently, then:

$$n(\circ) = \underset{?c}{\text{argmax}}\texttt{<l>}\{l \mid \begin{pmatrix} \texttt{<l>} & \texttt{a stig:Location .} \\ \texttt{<p>} & \texttt{a trucks:Package ;} \\ & \texttt{stig:adjacentTo <l> ;} \\ & \texttt{stig:carries [a stig:Stigma ;} \\ & \texttt{stig:level ?c] .} \end{pmatrix} \in G, c > 0\} \quad (42)$$

Consequently, the truck moves to the location with the package with the highest marker concentration c.

4. If for none of the packages it holds that marker concentration $c > 0$, move to the location where the package in the front of the loading bay, i.e., the package that would be unloaded next, needs to be delivered:

$$n(\circ) = \underset{?pos}{\text{argmin}}\texttt{<l>}\{l \mid \begin{pmatrix} \texttt{<p>} & \texttt{stig:adjacentTo <a> .} \\ \texttt{<a>} & \texttt{a trucks:Area;} \\ & \texttt{trucks:position ?pos .} \\ \texttt{<g>} & \texttt{a trucks:Goal ;} \\ & \texttt{trucks:payload <p> ;} \\ & \texttt{trucks:destination <l> .} \end{pmatrix} \in G\} \quad (43)$$

5. Finally, if the truck is empty and none of the packages in any location carries a marker with marker concentration $c > 0$:

$$n(\circ) = \underset{|p|}{\text{argmax}_{\texttt{<l>}}}\{l \mid \begin{pmatrix} \texttt{<l>} & \texttt{a trucks:Location .} \\ \texttt{<p>} & \texttt{a trucks:Package ;} \\ & \texttt{stig:adjacentTo <l> .} \end{pmatrix} \tag{44}$$

By the behaviour as defined above, `TRUCK` agents react to the stigmata generated and updated during evolution of the environment, as described in Section 6.2, or the relation between *package* resources, and other Topoi. With marker concentrations being calculated before queries by the truck agent are evaluated against the environment state, as it is defined for environment evolution in Equation (20), the truck agent always perceives the accurate marker concentrations, at the relevant topoi, at time of perception.

6.4. Evaluation

We evaluate above agents against Problems 1 through 5 of the time constrained trucks world challenge of the IPC-5 (The original problem definitions are available for download from https://lpg.unibs.it/ipc-5/, accessed on 21 March 2022, under "Resources"). These problems each provide scenarios with a single truck agent. An extension of our trucks world model to scenarios with a number of trucks operating simultaneously is ongoing, and publication of the results is subject to future work. To show the adaptability of the system and the chosen agent model, we use the same agent model and implementation throughout every problem instance, without further adaption towards individual challenges.

Table 2 shows the problem dimensions of the different problem instances: Each of the problems accommodates a set of 3 interconnected locations each, with an increasing number of packages that need to be delivered. In all but the second problem instance, 3 of the delivery instructions are constrained by a deadline, with the second problem specifying only one deadline.

Table 2. Problem sizes of the Trucks World single agent problem instances.

	Problem 1	**Problem 2**	**Problem 3**	**Problem 4**	**Problem 5**
# locations	3	3	3	3	3
# Packages	3	4	5	6	7
# Deadlines	3	1	3	3	3

The results of the evaluation are shown in Tables 3 and 4. Table 3 shows the deadlines defined by the individual problem instances, and the time by which the respective packages were delivered to the defined target destinations. Clearly, the agent manages to coordinate the pickup and delivery runs in every problem to meet each given deadline.

Table 3. Deadlines and times of deliveries per package in the different problem instances: the truck agent manages to meet all deadlines in all problems.

		Problem 1	**Problem 2**	**Problem 3**	**Problem 4**	**Problem 5**
Pack. 1	Deadline	919.7	842.7	616.7	537.3	992.8
	Delivered	432.9	769.1	569.6	491.4	729.2
Pack. 2	Deadline	919.7	N/A	925.1	1026.9	1866.7
	Delivered	842.2	N/A	283.3	942.5	292.6
Pack. 3	Deadline	1813.7	N/A	925.1	2878.2	2878.0
	Delivered	844.2	N/A	285.3	1737.2	2255.8

Table 4. Number of steps and make span taken until the last package was delivered by the compared approaches.

		MIPS-XXL	SGPLAN	stigLD
Problem 1	Steps	12	14	12
	MakeSpan	845.32	845.23	844.20
Problem 2	Steps	17	17	17
	MakeSpan	1714.57	1711.44	1713.4
Problem 3	Steps	19	19	19
	MakeSpan	1474.29	1470.14	1473.1
Problem 4	Steps	23	23	23
	MakeSpan	2634.63	2629.45	2676.7
Problem 5	Steps	N/A	28	29
	MakeSpan	N/A	1870.06	2255.8

Table 4 compares the performance of our approach to the two planners that solved the time constraint trucks world problem in the IPC-5 (Results are available for download from https://lpg.unibs.it/ipc-5/, accessed on 21 March 2022, under "Resources"), MIPS-XXL [42] and SGPLAN [43]. This comparison shows that our solution does not only find valid solutions in every problem instance w.r.t. deadlines, but also performs in the range of efficiency as the reference planners. In all of the problems, we find solutions in at most as many steps as the reference planners, while delivering all packages in the same time ranges as the reference planners. Only in Problem 5, our solution takes one step more than the only other competitor that solved the problem, SGPLAN, whereas MIPS-XXL did not provide a solution at all.

Time Constraint Trucks World with Disturbances

A main feature of stigmergy-based systems is their robustness against disturbances in the environment [5]. In order to demonstrate that by employing the methods and technologies from this paper, a robust, adaptive agent behaviour emerges, we further extended the original IPC trucks challenge.

In our extension, after a given time, new packages are created at a given locations, and with given deadlines. Deadlines of packages that had not been delivered at the time of disturbance are increased (coloured blue in Table 5) to keep the problem satisfiable, even if the truck has to pick up newly created packages before those originally in the problem. We omitted problems 1 and 2, as the low number of packages in the original problems did not leave much of a margin to introduce disturbances that would actually disturb the original problem execution.

Table 5. Adapted deadlines and time of completion for packages after introduced disturbance at time *t*. Red: Newly added packages with deadlines, Blue: Deadlines changed from the original problem.

	t	Pack 1	Pack 2	Pack 3	Pack 4	Pack 5	
Problem 3	300	925.1	925.1	1792.58	1744.48	1744.48	Deadline
		283.3	285.3	569.6	1136.2	1721.2	Delivered
Problem 4	950	537.3	1026.9	3629.56	4224.76	4224.76	Deadline
		491.4	942.5	1737.2	2979.0	4221.8	Delivered
Problem 5	500	1866.7	1877.32	3762.52	1393.42	1393.42	Deadline
		292.6	1783.8	2692.4	822.4	1349.2	Delivered

Time of disturbances and new deadlines are chosen such that the disturbance happens before the plan for the original problem has finished, i.e., the truck agent will have to react to the newly created package delivery orders before finishing delivery of the packages that were originally in the problem.

New packages are created on the same location, but are to be delivered to different destinations. We calculated the new deadlines such that while the truck may take one connection between locations twice to deliver each of the packages to a different destination, it is left only one movement to a neighbouring location, leading to that the truck will have to combine deliveries to make the new deadlines. Remaining deadlines after the newly introduced deadlines are shifted accordingly.

The results of our experiments with induced disturbances is shown in Table 5. Despite constraints changing while the simulation is already running, the truck manages to meet any given deadline, old ones as well as new ones. In Problem 5, the new deadlines were created after the truck has already delivered the first package, thus deadline and delivery time of that package did not change compared to the original problem. However, as soon as new deadlines are generated, the truck prefers the newly created packages with stricter deadlines over the originally placed ones (see Problem 5). Given that none of the agents maintain memory or plans, we can by this show that the presented system is able to react dynamically to changes in the environment, and without (re-)planning, manages to generate valid solutions in dynamic environments.

7. Conclusions and Future Work

We proposed 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 an extension mechanism for specifying the *domain-specific* hypermedia-driven behaviour of Linked Data agents.

Thereafter, we investigated stigmergic coordination principles, and their implementation within such Linked Systems. When considering some of the prime examples of stigmergy, however, (e.g., ant or termite colony optimisation methods [37,38]), it became apparent that a purely reactive Linked Data environment is insufficient for the implementation of transient marker-based stigmergic systems. In fact, the environments of such stigmergic systems typically demonstrate some endogenous dynamics, e.g., diffusion, evaporation, or atrophy, and - in addition to being malleable and perceivable by all agents under coordination—*actively* drive the evolution of such agent-less dynamic processes. Based on this observation, we developed `stigLD` (https://github.com/dfki-asr/stigLD, accessed on 21 March 2022), a Linked Data framework for facilitating the design and declarative implementation of sematectonic, persistent marker-based and transient marker-based stigmergic mechanisms within a hypermedia MAS.

We demonstrated the genericity and effectiveness of our modelling approach by two evaluation scenarios. First, in a make-to-order (MTO) scenario from the production domain, we demonstrated how to employ the `stigLD` concepts and domain model in order to achieve self-coordination in a multi agent system using two transient semio-chemical marker models. Our implementation displayed emergence of self-organised coordination from simple agent behaviour and compared favourably against a random walk baseline strategy. Second, we implemented a stigmergic agent system to solve instances of the (time constrained) Trucks World domain as presented in the International Planning Competition 5 (IPC-5) [8]. Despite its simplicity, our implementation solved all problem instances without constraint violations and is on-par with the leading planners in IPC-5.

The current state of the presented work requires to transfer the agent programs from CCS specifications to an executable form, as CCS itself is not directly executable. We are for this evaluating Behavior Trees [44,45] as semantically equivalent, but executable representation of CCS expressions.

In ongoing research, we are concerned with bench-marking against more problems from the International Planning Competition [8] and investigate evolutionary approaches for automating the design of stigmergic hypermedia MASs.

Author Contributions: Conceptualization, R.S., T.S. and M.C.; Formal analysis, R.S. and T.S.; Funding acquisition, R.S.; Investigation, R.S., T.S. and M.C.; Methodology, R.S., T.S. and M.C.; Software, T.S. and M.C.; Supervision, R.S.; Validation, T.S. and M.C.; Visualization, M.C.; Writing—original draft, R.S., T.S. and M.C.; Writing—review & editing, R.S., T.S. and M.C. All authors have read and agreed to the published version of the manuscript.

Funding: 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).

Data Availability Statement: The data presented in this study are openly available in Github at https://github.com/dfki-asr/BMBF-MOSAIK, accessed on 21 March 2022.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. 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 Conference on Autonomous Agents and Multiagent Systems, Auckland, New Zealand, 9–13 May 2019.
2. Ciortea, A.; Boissier, O.; Ricci, A. *Engineering World-Wide Multi-Agent Systems with Hypermedia*; Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Berlin/Heidelberg, Germany, 2019; Volume 11375 LNAI, pp. 285–301. [CrossRef]
3. Harth, A.; Käfer, T. Towards Specification and Execution of Linked Systems. In Proceedings of the 28th GI-Workshop Grundlagen von Datenbanken (GvD), Nörten-Hardenberg, Germany, 24–27 May 2016; pp. 62–67.
4. 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, Rio de Janeiro, Brazil, 13–17 May 2008; pp. 1265–1266.
5. Heylighen, F. Stigmergy as a universal coordination mechanism I: Definition and components. *Cogn. Syst. Res.* **2016**, *38*, 4–13. [CrossRef]
6. Heylighen, F. Stigmergy as a universal coordination mechanism II: Varieties and evolution. *Cogn. Syst. Res.* **2016**, *38*, 50–59. [CrossRef]
7. Spieldenner., T.; Chelli., M. Linked Data as Stigmergic Medium for Decentralized Coordination. In Proceedings of the 16th International Conference on Software Technologies—ICSOFT, Online Streaming, 6–8 July 2021; pp. 347–357. [CrossRef]
8. Gerevini, A.E.; Haslum, P.; Long, D.; Saetti, A.; Dimopoulos, Y. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.* **2009**, *173*, 619–668. [CrossRef]
9. Heylighen, F.; Vidal, C. Getting things done: The science behind stress-free productivity. *Long Range Plan.* **2008**, *41*, 585–605. [CrossRef]
10. Tummolini, L.; Castelfranchi, C. *Trace Signals: The Meanings of Stigmergy*; Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Berlin/Heidelberg, Germany, 2007; Volume 4389, pp. 141–156. [CrossRef]
11. Theraulaz, G.; Bonabeau, E. A brief history of stigmergy. *Artif. Life* **1999**, *5*, 97–116. [CrossRef] [PubMed]
12. Dipple, A.; Raymond, K.; Docherty, M. *General Theory of Stigmergy: Modelling Stigma Semantics*; Elsevier: Amsterdam, The Netherlands, 2014. [CrossRef]
13. Hadeli, K.; Valckenaers, P.; Zamfirescu, C.; Van Brussel, H.; Saint Germain, B.; Hoelvoet, T.; Steegmans, E. Self-organising in multi-agent coordination and control using stigmergy. In Proceedings of the International Workshop on Engineering Self-Organising Applications, Melbourne, Australia, 15 July 2003; pp. 105–123.
14. Hadeli, K.; Valckenaers, P.; Kollingbaum, M.; Van Brussel, H. Multi-agent coordination and control using stigmergy. *Comput. Ind.* **2004**, *53*, 75–96. [CrossRef]
15. Valckenaers, P.; Van Brussel, H.; Kollingbaum, M.; Bochmann, O. Multi-agent Coordination and Control Using Stigmergy Applied to Manufacturing Control. In *Multi-Agent Systems and Applications, Proceedings of the 9th ECCAI Advanced Course, ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001, Prague, Czech Republic, 2–13 July 2001*; Selected Tutorial Papers; Luck, M.; Mařík, V., Štěpánková, O., Trappl, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 317–334. [CrossRef]
16. Krieger, M.J.; Billeter, J.B.; Keller, L. Ant-like task allocation and recruitment in cooperative robots. *Nature* **2000**, *406*, 992–995. [CrossRef] [PubMed]
17. Jevtić, A.; Gutierrez, Á.; Andina, D.; Jamshidi, M. Distributed bees algorithm for task allocation in swarm of robots. *IEEE Syst. J.* **2012**, *6*, 296–304. [CrossRef]
18. Kanamori, R.; Takahashi, J.; Ito, T. Evaluation of traffic management strategies with anticipatory stigmergy. *J. Inf. Process.* **2014**, *22*, 228–234. [CrossRef]

19. 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.* **2018**, *19*, 2258–2267. [CrossRef]
20. Weyns, D.; Holvoet, T. *Model for Simultaneous Actions in Situated Multi-Agent Systems*; Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science); Springer: Berlin/Heidelberg, Germany, 2003; Volume 2831, pp. 105–118. [CrossRef]
21. Weyns, D.; Holvoet, T. A formal model for situated multi-agent systems. *Fundam. Inform.* **2004**, *63*, 125–158.
22. 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. Agents-Multi-Agent Syst.* **2007**, *14*, 5–30. [CrossRef]
23. Ferber, J.; Müller, J.P. Influences and Reaction: A Model of Situated Multiagent Systems. In Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS-96), Kyoto, Japan, 10–13 December 1996; pp. 72–79.
24. Ciortea, A.; Mayer, S.; Boissier, O.; Gandon, F. Exploiting Interaction Affordances: On Engineering Autonomous Systems for the Web of Things. In Proceedings of the Second W3C Workshop on the Web of Things: The Open Web to Challenge IoT Fragmentation, Munich, Germany, 3—5 June 2019.
25. Hunt, E.R.; Jones, S.; Hauert, S. Testing the limits of pheromone stigmergy in high-density robot swarms. *R. Soc. Open Sci.* **2019**, *6*, 190225. [CrossRef] [PubMed]
26. Jochum, B.; Nürnberg, L.; Aßfalg, N.; Käfer, T. *Data-Driven Workflows for Specifying and Executing Agents in an Environment of Reasoning and RESTful Systems*; Lecture Notes in Business Information Processing; Springer: Berlin/Heidelberg, Germany, 2019; Volume 362, pp. 93–105. [CrossRef]
27. 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, Hyderabad, India, 28 March–1 April 2011; pp. 355–360.
28. Privat, G. Phenotropic and stigmergic webs: The new reach of networks. *Univers. Access Inf. Soc.* **2012**, *11*, 323–335. [CrossRef]
29. Charpenay, V.; Schraudner, D.; Seidelmann, T.; Spieldenner, T.; Weise, J.; Schubotz, R.; Mostaghim, S.; Harth, A. MOSAIK: A Formal Model for Self-Organizing Manufacturing Systems. *IEEE Pervasive Comput.* **2020**, *20*, 9–19. [CrossRef]
30. Milner, R. *A Calculus of Communicating Systems*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1980; Volume 92. [CrossRef]
31. Milner, R. *Communication and Concurrency*; PHI Series in Computer Science; Prentice Hall: Hoboken , NJ, USA, 1989.
32. Genesereth, M.R.; Nilsson, N.J. *Logical Foundations of Artificial Intelligence*; Morgan Kaufmann: Burlington, MA, USA, 2012.
33. Smith, G.J.; Gero, J.S. What does an artificial design agent mean by being 'situated'? *Des. Stud.* **2005**, *26*, 535–561. [CrossRef]
34. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1996**, *26*, 29–41. [CrossRef] [PubMed]
35. Dorigo, M.; Di Caro, G. Ant colony optimization: A new meta-heuristic. In Proceedings of the 1999 Congress on Evolutionary Computation, Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1470–1477. [CrossRef]
36. Dorigo, M.; Bonabeau, E.; Theraulaz, G. Ant algorithms and stigmergy. *Future Gener. Comput. Syst.* **2000**, *16*, 851–871. [CrossRef]
37. Dorigo, M.; Stützle, T. Ant colony optimization: Overview and recent advances. In *International Series in Operations Research and Management Science*; Springer: New York, NY, USA, 2019; Volume 272, pp. 311–351. [CrossRef]
38. Hedayatzadeh, R.; Akhavan Salmassi, F.; Keshtgari, M.; Akbari, R.; Ziarati, K. Termite colony optimization: A novel approach for optimizing continuous problems. In Proceedings of the 2010 18th Iranian Conference on Electrical Engineering, Isfahan, Iran, 11–13 May 2010; pp. 553–558. [CrossRef]
39. Van Dyke Parunak, H. A Survey of Environments and Mechanisms for Human-Human Stigmergy. In *Environments for Multi-Agent Systems II*; Weyns, D., Van Dyke Parunak, H., Michel, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 163–186.
40. Fielding, R.T.; Reschke, J. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231. 2014. Available online: https://www.rfc-editor.org/info/rfc7231 (accessed on 21 March 2022).
41. Fielding, R. Re: Draft Findings on Unsafe Methods (whenToUseGet-7). 2002. Online. Available online: https://lists.w3.org/Archives/Public/www-tag/2002Apr/0150.html (accessed on 21 March 2022).
42. Edelkamp, S.; Jabbar, S.; Nazih, M. Large-scale optimal PDDL3 planning with MIPS-XXL. In Proceedings of the 5th International Planning Competition Booklet (IPC-2006), Hakodate, Japan, 8 May 2006; pp. 28–30.
43. Chen, Y.; Hsu, C.W.; Wah, B.W. SGPlan: Subgoal partitioning and resolution in planning. In Proceedings of the 4th International Planning Competition (IPC4), Hosted at the International Conference on Automated Planning and Scheduling, ICAPS'04, Whistler, BC, Canada, 3–7 June 2004 ; pp. 30–33 .
44. Colledanchise, M.; Ögren, P. *Behavior Trees in Robotics and AI: An Introduction*; CRC Press: Boca Raton, FL, USA, 2018.
45. Shoulson, A.; Garcia, F.M.; Jones, M.; Mead, R.; Badler, N.I. Parameterizing behavior trees. In Proceedings of the International Conference on Motion in Games, Edinburgh, UK, 13–15 November 2011; pp. 144–155.