Dynamics in Information Systems: Specification, Construction, and Correctness

—An Algebraic Approach—

Ingo Claßen,[1] Martin Gogolla,[2] and Michael Löwe[1]

[1] Technical University of Berlin
[2] University of Bremen

Abstract. A formal framework based on algebraic graph theory is presented that integrates specification and construction of dynamics in information systems. Specifications are based on temporal logic whose semantics is given by algebras and partial homomorphisms. Constructions are given by graph transformation rules whose operational nature provides a first step towards actual implementations. Both are related by a correctness notion.
The formal framework is especially suited as a semantical basis for graphical notations as used in conceptual modeling, thus combining intuitiveness of such notations with precision of formal methods.

1 Introduction

Conceptual modeling, i.e., specification of information systems and database applications is traditionally done by employing semantic data models [26, 27, 15, 23]. Although structure and behavior of information systems must be taken into account, conceptual modeling classically focuses on the structural aspects only, for instance by using the well-tried Entity-Relationship model [5] and its numerous extensions [10, 1, 11]. In order to treat information systems behavior, temporal logic [22] has been proposed [13, 9, 17]. More recently, modern object-oriented software development techniques like OOA/OOD [4], OMT [25] or the use case driven approach in [16] combine structural and behavioral aspects by an informal integration of class or object diagrams (for the structural part) with extensions of state machine formalisms (for the behavioral part) often following the statecharts approach [14].

The success of conceptual modeling methods is mainly due to the intuitive graphical notations they employ. Therefore it is natural to provide a semantical basis for such notations to combine the intuitiveness of graphics with the precision of formal methods. We provided such kind of semantics based on algebraic graph theory in [7, 8]. One of its advantages is that it provides means to describe operations (transactions) also on the graphical level in terms of graph transformation rules. Thus, the intuitiveness of structural aspects is extended to dynamic aspects. The employed algebraic approach provides a natural mapping of graphical notation to the underlying semantical domain and a rich theory on the operational side.

The aim of this paper is to complement the operational approach of our previous work (based on algebraic graph transformation rules) by a descriptive component. For this reason we introduce a temporal logic and a corresponding semantical interpretation that is adapted to the domain of algebraic structures. In this way we try to achieve a formal framework that integrates structural and dynamic aspects of information systems on the specification as well as on the construction level. The integration is supported by a formal correctness relation between specifications and constructions.

The material in the following sections concentrates on the technical foundation and not so much on notational convenience. The proposed framework is not intended to be used directly in the development of information systems but as a semantical basis for a user–friendly conceptual modeling technique.

The paper is organized as follows. Sect. 2 introduces the running example (in a suitable graphical notation) we are using throughout the paper. Sect. 3 introduces our temporal logic and its semantical domain in terms of algebras and homomorphisms. This logic is adapted in Sect. 4 to the specification of information systems whose structure is defined in a graphical way. Sect. 5 defines the construction of information systems and in Sect. 6 suitable correctness notions are discussed. The concluding remarks in Sect. 7 describe some open questions for future research.

2 Motivating Example

The running example of this paper is a small database system that supports the handling of the formal aspects of examinations at universities (i.e. registration, sitting, deregistration, etc.). It relies on the information scheme[3] presented in Fig. 1 that describes the static information structure of possible instances (e.g. database states). The main scenario we are going to handle in a formal way deals with the registration of examinations.

Here we make the assumption that after the registration of an examination a student has a time limit to actually sit it, determined by attribute `lastDay`. The time range for an examination begins at the day of the registration (given by attribute `systemDate`) and ends a number of days (determined by attribute `timeUntilExam`) after the registration. The actual day of the examination is stored in attribute `day` and the result in attribute `passed`. After a student has registered himself for an examination (only in this case attribute `lastDay` is defined) the scenario can proceed in three alternative ways:

1. The student sits the examination in question within the given time limit. Then attributes `day` and `passed` are set accordingly and attribute `lastDay` gets undefined again.

[3] The scheme uses a graphical notation as given e.g. in Rumbaugh and is essentially a variant of an entity–relationship diagram. To keep the example small, we only have introduced classes and attributes that are used later–on. The star (*) after `lastDay`, `day`, and `passed` denotes optional attributes.

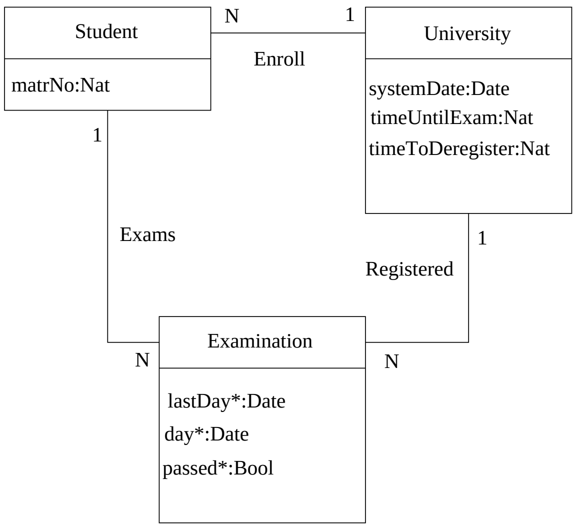

Fig. 1. Information scheme

2. The student deregisters the examination. This is only possible within the time limit determined by attribute `timeToDeregister` (e.g. at most three days before `lastDay`).
3. The examination does not take place until `lastDay`. Then the examination is regarded as sat but with negative result. This means that the attributes are handled like in case 1., but in some automatic way.

3 Algebraic Semantics for Temporal Logic

This section defines syntax and semantics of our temporal logic in a general setting, where the syntactical level is a variant of first order logic enriched by temporal operators like **always**, **sometime**, etc. (see [22]), and the semantical level is given by many–sorted total algebras and partial homomorphisms.

The general setting will not be used directly but will be adapted to graphical notations for the specification of information systems as given in Fig. 1. Therefore, the formal interpretation of our running example (Sect. 2) is deferred until we have introduced transition signatures in Sect. 4.

Definition 1 (Temporal Formulas). Let $\Sigma = (S, \Omega)$ be an algebraic signature and $X = (X_s)_{s \in S}$ a family of variables. The set of temporal formulas $\mathcal{TF}(\Sigma, X)$ is inductively defined as follows:

1. Atomic Formulas:
   $t_1 = t_2 \in \mathcal{TF}(\Sigma, X)$ for all $t_1, t_2 \in T_\Sigma(X)_s$

2. Junctors: Let φ and ψ be temporal formulas. Then
   (a) $\neg\varphi$
   (b) $\varphi \vee \psi$
   are also temporal formulas.
3. Quantification: Let φ be a temporal formula and $x \in X_s$ a variable. Then
   $\exists(x)\varphi$
   is also a temporal formula.
4. Temporal Quantification: Let φ be a temporal formula. Then
   sometime φ
   is also a temporal formula.

As usual, other junctors and quantifiers can be regarded as short–hands, e.g. $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$, $\forall(x)\varphi \equiv \neg\exists(x)\neg\varphi$, **always** $\varphi \equiv \neg$**sometime** $\neg\varphi$, and so on.

On the semantical level we use many–sorted total algebras as states (see [18]) and partial homomorphisms as state transitions. The latter provides an explicit relation between state elements before and after transitions. Partiality of homomorphisms is needed to deal with deletion of objects.

Definition 2 (Category $\mathrm{ALG}^p(\Sigma)$). Let $\Sigma = (S, \Omega)$ be an algebraic signature. The objects of $\mathrm{ALG}^p(\Sigma)$ are given by total Σ–algebras. A morphism $f: A \to B$ in $\mathrm{ALG}^p(\Sigma)$ is given by a span $(A \overset{i}{\hookleftarrow} \mathrm{dom}(f) \overset{f}{\rightarrow} B)$, where $\mathrm{dom}(f)$ is a subalgebra of A, i is the inclusion from $\mathrm{dom}(f)$ in A, and $f: \mathrm{dom}(f) \to B$ is a total homomorphism between Σ–algebras.[4] Composition of $(A \overset{i}{\hookleftarrow} \mathrm{dom}(f) \overset{f}{\rightarrow} B)$ and $(B \overset{j}{\hookleftarrow} \mathrm{dom}(g) \overset{g}{\rightarrow} C)$ is given by $(A \overset{i \circ j'}{\hookleftarrow} \mathrm{dom}(g \circ f) \overset{g \circ f'}{\rightarrow} C)$

$$\begin{tikzcd}
A & & B & & C \\
& \mathrm{dom}(f) \arrow[ul, "i"] \arrow[ur, "f"'] & & \mathrm{dom}(g) \arrow[ul, "j"'] \arrow[ur, "g"'] & \\
& & \mathrm{dom}(g \circ f) \arrow[ul, "j'"] \arrow[ur, "f'"'] & &
\end{tikzcd}$$

where $\mathrm{dom}(g \circ f)$, j', and f' are given by the inverse image of f and j. Identities are given by $(A \overset{1_A}{\hookleftarrow} \mathrm{dom}(1_A) \overset{1_A}{\rightarrow} A)$ where 1_A is the identity on Σ–algebra A.[5]

Note that $\mathrm{dom}(1_A) = A$.

Definition 3 (Partial Assignment, Evaluation). Let $\Sigma = (S, \Omega)$ be an algebraic signature, A a Σ–algebra, $X = (X_s)_{s \in S}$ a family of variables, and

[4] The interpretation of state transitions as spans is closely related to the notion of state transition in the algebraic semantics of the object specification language TROLL light [12].

[5] Actually, our notion of (partial) morphism corresponds to *one* span and not to an isomorphism class of spans as is usual in the categorical treatment of such morphisms (see, e.g. [2]). Of course, our work can also be extended to this more complexer notion.

$t \in T_{\Sigma}(X)$. A partial assignment $a^p : X \to A$ is a family $(a^p_s : X_s \to A_s)_{s \in S}$ of partial mappings. The set $\mathrm{var}(t) \subseteq X$ denotes the variables actually occurring in t and $\mathrm{dom}(a^p) \subseteq X$ denotes the set of variables for which a^p is defined. The evaluation $\overline{a^p}(t)$ is defined if $\mathrm{var}(t) \subseteq \mathrm{dom}(a^p)$. In this case the value is given by the usual evaluation of terms.

Definition 4 (Sequential Run). A sequential run $\underline{A}$ in $\mathrm{ALG}^p(\Sigma)$ is a functor $\underline{A}: I \to \mathrm{ALG}^p(\Sigma)$ where $(I, \to)$ is linear order isomorphic to $(N, \leq)$, i.e. $(I, \to)$ has the following structure:[6]

$$i_0 \to i_1 \to \cdots \to i_n \to \cdots$$

Finite sequential runs are special sequential runs where there is an $i \in I$ such that $\underline{A}(i \to j) = 1_{\underline{A}(i)}$ for all $i \to j$. The sequential run $\underline{A}_i : I_i \to \mathrm{ALG}^p(\Sigma)$, $i \in I$, is given by cutting off all states before i from $\underline{A}$, i.e. $\underline{A}_i$ is defined as follows: I_i is the full subcategory of I containing all objects $j \in I$ with $i \to j$ and $\underline{A}_i = \underline{A}|_{I_i}$, i.e. the functor $\underline{A}$ restricted to the subcategory I_i.

Definition 5 (Run Assignment). Let $\underline{A}: I \to \mathrm{ALG}^p(\Sigma)$ be a sequential run, and X a family of variables. A run assignment $\underline{a}^p : X \to \underline{A}$ is a family $(\underline{a}^p(i) : X \to \underline{A}(i))_{i \in I}$ of partial assignments such that for all $i \to j \in I$ the following holds:

$$x \in \mathrm{dom}(\underline{a}^p(i)) \Longrightarrow \underline{A}(i \to j)(\underline{a}^p(i)(x)) = \underline{a}^p(j)(x)^7$$

This means that if x is in the domain of $\underline{a}^p(i)$ the following must hold for $\underline{a}^p(j)$: If $\underline{a}^p(i)(x)$ is mapped by $\underline{A}(i \to j)$, then x must also be in the domain of $\underline{a}^p(j)$ and must have a compatible assignment, i.e. $\underline{A}(i \to j)(\underline{a}^p(i)(x))$ and $\underline{a}^p(j)(x)$ must be the same elements. If $\underline{a}^p(i)(x)$ is not mapped by $\underline{A}(i \to j)$, i.e. it is removed during the transition from $\underline{A}(i)$ to $\underline{A}(j)$ then x is must not be in the domain of $\underline{a}^p(j)$.

The run assignment $\underline{a}^p_i$ wrt. $\underline{a}^p$ is defined in the same way as $\underline{A}_i$ wrt. $\underline{A}$, i.e. $\underline{a}^p_i = (\underline{a}^p(j) : X \to \underline{A}(j))_{j \in I_i}$.

Lemma 6. *Let $\underline{A}, \underline{B}: I \to \mathrm{ALG}^p(\Sigma)$ be two sequential runs, $\underline{a}^p : X \to \underline{A}$ a run assignment and $\tau: \underline{A} \Rightarrow \underline{B}$ a natural isomorphism. Then $\tau \circ \underline{a}^p : X \to \underline{B}$ is also a run assignment.*

Proof. Let $i \to j \in I$. Then the following diagram immediately shows the required compatibility of Def. 5. Note that (1) commutes on the domain of $\underline{a}^p(i)$ and $\tau(i)$ and $\tau(j)$ must be total since they are isomorphisms in $\mathrm{ALG}^p(\Sigma)$. Thus the diagram commutes on the domain of $\tau(i) \circ \underline{a}^p(i)$.

[6] A partial order $(I, \to)$ can be regarded as a category with object set I and $i \to j$ as the only morphism between i and j, if $(i, j) \in \to$.

[7] Note that $=$ denotes the following equality: Either both sides are defined and have same value or both sides are undefined (often denoted by $=_\perp$).

$$\begin{tikzcd}
 & X \arrow[dl, "\underline{a}^p(i)"'] \arrow[dr, "\underline{a}^p(j)"] \arrow[d, phantom, "(1)"] & \\
\underline{A}(i) \arrow[rr, "\underline{A}(i\to j)"] \arrow[d, "\tau(i)"] \arrow[drr, phantom, "(=)"] & & \underline{A}(j) \arrow[d, "\tau(j)"] \\
\underline{B}(i) \arrow[rr, "\underline{B}(i\to j)"] & & \underline{B}(j)
\end{tikzcd}$$

Definition 7 (Satisfaction of Temporal Formulas). Satisfaction of a temporal formula φ in a sequential run $\underline{A}$ under a run assignment $\underline{a}^p : X \to \underline{A}$, short $[\underline{A}, \underline{a}^p] \models \varphi$, is inductively defined by:

1. Atomic Formulas:
   $[\underline{A}, \underline{a}^p] \models t_1 = t_2$ iff $\mathrm{var}(t_1) \cup \mathrm{var}(t_2) \subseteq \mathrm{dom}(\underline{a}^p(0))$ and $\overline{\underline{a}^p(0)}(t_1) = \overline{\underline{a}^p(0)}(t_2)$[8]
2. Junctors:
   (a) $[\underline{A}, \underline{a}^p] \models \neg\varphi$ iff not $[\underline{A}, \underline{a}^p] \models \varphi$
   (b) $[\underline{A}, \underline{a}^p] \models \varphi \vee \psi$ iff $[\underline{A}, \underline{a}^p] \models \varphi$ or $[\underline{A}, \underline{a}^p] \models \psi$
3. Quantification:[9]
   $[\underline{A}, \underline{a}^p] \models \exists(x)\varphi$ iff there is a run assignment $\underline{b}^p : X \to \underline{A}$ with $\underline{b}^p(i)|_{X\setminus\{x\}} = \underline{a}^p(i)|_{X\setminus\{x\}}$ for all $i \in I$ such that $[\underline{A}, \underline{b}^p] \models \varphi$[10]
4. Temporal Quantification:
   $[\underline{A}, \underline{a}^p] \models$ **sometime** φ iff there exists an $i \in I$ such that $[\underline{A}_i, \underline{a}^p_i] \models \varphi$

A sequential run $\underline{A}$ satisfies a temporal formula φ, short $\underline{A} \models \varphi$, if $[\underline{A}, \underline{a}^p] \models \varphi$ for all run assignments $\underline{a}^p$.

The following proposition, lemma and corollary show that satisfaction of formulas is invariant under isomorphism. This property is essential since construction of information systems as presented in Sect. 5 only provides results up to isomorphisms.

Proposition 8 (Invariance under Isomorphism). *Let $\underline{A}, \underline{B} : I \to \mathrm{ALG}^p(\Sigma)$ be two sequential runs, $\varphi \in \mathcal{TF}(\Sigma, X)$ a temporal formula, $\tau : \underline{A} \Rightarrow \underline{B}$ a natural isomorphism, $\underline{a}^p : X \to \underline{A}$ a run assignment, and $i \in I$. Then*

$$[\underline{A}_i, \underline{a}^p_i] \models \varphi \Longleftrightarrow [\underline{B}_i, (\tau \circ \underline{b}^p)_i] \models \varphi$$

Proof. By induction on the structure of formulas (see Def. 1). We only show $[\underline{A}_i, \underline{a}^p_i] \models \varphi \Longrightarrow [\underline{B}_i, (\tau \circ \underline{b}^p)_i] \models \varphi$ in each of the following cases. The other direction follows from application of Lemma 9.

---

[8] Here 0 denotes the least element of the given order $(I, \to)$.

[9] We assume that all bound variables in one formula have different names.

[10] $\underline{a}^p(i)|_{X\setminus\{x\}}$ denotes the restriction of $\underline{a}^p$ to $X \setminus \{x\}$ and $=$ is the equality of partial functions, i.e. both functions have same domain and lead to same results on all elements they are defined on.

1. $\varphi \equiv t_1 = t_2$
According to Def. 7(1) we have $\mathrm{var}(t_1) \cup \mathrm{var}(t_2) \subseteq \mathrm{dom}(\underline{a}_i^p(0))$ and $\overline{\underline{a}_i^p(0)}(t_1) = \overline{\underline{a}_i^p(0)}(t_2)$ $(\star)$. Since $\tau(0)$ is an isomorphism in $\mathrm{ALG}^p(\Sigma)$ it must be total. Therefore $\mathrm{dom}((\tau \circ \underline{a}^p)_i(0)) = \mathrm{dom}(\underline{a}_i^p(0))$, i.e. $\mathrm{var}(t_1) \cup \mathrm{var}(t_2) \subseteq \mathrm{dom}((\tau \circ \underline{a}^p)_i(0))$. Then
$$\begin{aligned}(\tau \circ \underline{a}^p)_i(0)(t_1) &= \\ &= (\tau(i)(\underline{a}_i^p(0)(t_1)) \\ &= (\tau(i)(\underline{a}_i^p(0)(t_2)) \\ &= (\tau \circ \underline{a}^p)_i(0)(t_2)\end{aligned}$$
2. $\varphi \equiv \neg\varphi'$, $\varphi \equiv \varphi_1 \vee \varphi_2$. We only show the case $\varphi \equiv \neg\varphi'$. The other case proceeds in the same way.
According to Def. 7(2) we have *not* $[\underline{A}_i, \underline{a}_i^p] \models \varphi'$. Induction hypothesis yields *not* $[\underline{B}_i, (\tau \circ \underline{a}^p)_i] \models \varphi'$. Using Def. 7(2) again we get $[\underline{B}_i, (\tau \circ \underline{a}^p)_i] \models \neg\varphi$.
3. $\varphi \equiv \exists(x)\varphi'$
According to Def. 7(3) there is an $\tilde{\underline{a}}^p : X \to \underline{A}_i$ with $[\underline{A}_i, \tilde{\underline{a}}^p] \models \varphi'$ where $\underline{a}_i^p$ and $\tilde{\underline{a}}^p$ only differ at variable x. Define $\underline{b}^p : X \to A$ as follows:
   (a) $\underline{b}^p(j)(y) = \underline{a}^p(j)(y)$ for all $y \neq x$ and all $j \to i, j \neq i$
   (b) $x \notin \mathrm{dom}(\underline{b}^p(j))$ for all $j \to i, j \neq i$
   (c) $\underline{b}_i^p = \tilde{\underline{a}}^p$
Since $\underline{b}_i^p = \tilde{\underline{a}}^p$, we immediately get $[\underline{A}_i, \underline{b}_i^p] \models \varphi'$. Induction hypothesis yields $[\underline{B}_i, (\tau \circ \underline{b}^p)_i] \models \varphi'$. Since $\tau \circ \underline{b}^p$ and $\tau \circ \underline{a}^p$ differ at most at variable x we get $[\underline{B}_i, (\tau \circ \underline{a}^p)_i] \models \exists(x)\varphi$.
4. $\varphi \equiv \textbf{sometime}\, \varphi'$
According to Def. 7(4) there is a $k \in I_i$ with $[(\underline{A}_i)_k, (\underline{a}_i^p)_k] \models \varphi'$. Since $i \to k$ we have $(\underline{A}_i)_k = \underline{A}_k$ and $(\underline{a}_i^p)_k = \underline{a}_k^p$. Induction hypothesis leads to $[\underline{B}_k, (\tau \circ \underline{a}^p)_k] \models \varphi'$. Using $i \to k$ again we get $[\underline{B}_i, (\tau \circ \underline{a}^p)_i] \models \textbf{sometime}\, \varphi$.

Lemma 9. *Let $\varphi \in \mathcal{TF}(\Sigma, X)$ be a fixed formula. If $[\underline{A}, \underline{a}^p] \models \varphi \Longrightarrow [\underline{B}, \tau \circ \underline{a}^p] \models \varphi$ holds for all sequential runs $\underline{A}, \underline{B}: I \to \mathrm{ALG}^p(\Sigma)$, natural isomorphisms $\tau: \underline{A} \Rightarrow \underline{B}$, and run assignments $\underline{a}^p: X \to \underline{A}$ then also $[\underline{B}, \tau \circ \underline{a}^p] \models \varphi \Longrightarrow [\underline{A}, \underline{a}^p] \models \varphi$ holds for all $\underline{A}$, $\underline{B}$, τ, and $\underline{a}^p$.*

Proof. Assume $[\underline{B}, \tau \circ \underline{a}^p] \models \varphi$ holds. Satisfaction of the premise (exchanging $\underline{A}$ and $\underline{B}$ and substituting $\tau^{-1} \circ \underline{a}^p$ for $\underline{a}^p$) leads to $[\underline{A}, (\tau \circ \tau^{-1} \circ \underline{a}^p)] \models \varphi$. This means $[\underline{A}, \underline{a}^p] \models \varphi$.

Corollary 10. *Let $\underline{A}, \underline{B}: I \to \mathrm{ALG}^p(\Sigma)$ be two sequential runs, $\varphi \in \mathcal{TF}(\Sigma, X)$ a temporal formula, and $\tau: \underline{A} \Rightarrow \underline{B}$ a natural isomorphism.*

1. *Let $\underline{a}^p: X \to \underline{A}$ be a run assignment. Then $[\underline{A}, \underline{a}^p] \models \varphi \Longleftrightarrow [\underline{B}, (\tau \circ \underline{a}^p)] \models \varphi$*
2. *$\underline{A} \models \varphi \Longleftrightarrow \underline{B} \models \varphi$*

Proof.

1. Use Prop. 8 with $i = 0$.
2. We only show $\underline{A} \models \varphi \Longrightarrow \underline{B} \models \varphi$. The opposite direction can be shown in an analogous way. Let $\underline{a}^p: X \to \underline{B}$ be an arbitrary run assignment. $\underline{A} \models \varphi$ implies $[\underline{A}, \tau^{-1} \circ \underline{a}^p] \models \varphi$. (1) in this corollary leads to $[\underline{B}, \tau \circ \tau^{-1} \circ \underline{a}^p] \models \varphi$, i.e. $[\underline{B}, \underline{a}^p] \models \varphi$.

4 Specification

While Sect. 3 presented a temporal logic and the corresponding satisfaction relation wrt. an arbitrary signature Σ, this section provides suitable restrictions on signatures and associated algebras that lead to a meaningful interpretation for the specification of information systems. One crucial aspect is a clear distinction between objects and values (see e.g. [3]); the former may be created, destroyed, and modified whereas the latter are assumed to be unchangeable. A second aspect pertains to the graphical structure of schemes as given in Fig. 1.

In [7] we have introduced transition signatures[11] and associated algebras as a flexible semantical basis for graphical specification techniques like entity–relationship models or object models. A transition signature is a structured algebraic signature introducing a syntactical separation between objects and values.

Definition 11 (Transition Signature). A transition signature

$$\Sigma = (\Sigma^G, \Sigma^D, \Omega^{attr})$$

consists of

- a graph signature $\Sigma^G = (S^G, \Omega^G)$, where Ω^G contains only unary operations,
- a data signature $\Sigma^D = (S^D, \Omega^D)$,
- and an $S^G \times S^D$–indexed family $\Omega^{attr} = (\Omega^{attr}_{s,s'})_{s \in S^G, s' \in S^D}$ of operation sets.

Σ can be regarded as a plain signature $\Sigma = (S, \Omega)$ by taking $S = S^G \cup S^D$ and $\Omega = \Omega^G \cup \Omega^D \cup \Omega^{attr}$.

Σ_G is called the *graph part* of Σ and is intended to represent objects and references.[12] Algebras with unary operations have a direct interpretation as graphs with typed nodes and edges. Moreover the restriction to unary operations is crucial to provide the semantical background for the construction of information systems in Sect. 5.

Σ_D is called the *data part*. Note that this is a plain signature without any restrictions on the kind of operations.

The last component of a transition signature is called the *attribute part*. It consists of unary operations from graph sorts to data sorts and provides attribute assignments, i.e. the association of attribute values to objects.

The information scheme in Fig. 1 leads to the transition signature given in Fig. 2. Transition signatures can be derived from graphical notations in a canonical way (see [7]). Note that all relationships and attributes are represented by sorts. The connection of relationships to classes is realized by operations,

[11] We have called them *attributed graph signatures* in that paper, but the name *transition signatures* is more suitable in the current context.

[12] The name graph part has been chosen, since a system of interrelated objects can be regarded as a graph.

trans Is
graph part **sorts** `Student`, `matrNo`
`Examination`, `lastday`, `day`, `passed`
`University`, `systemDate`, `timeUntilExam`
`timeToDeregister`
`Enroll`, `Exams`, `Registered`
opns `student`: `Enroll` $\to$ `Student`
`university`: `Enroll` $\to$ `University`
...
`matrNo-class`: `matrNo` $\to$ `Student`
...
data part **sorts** `Nat`, `Date`, `Bool`
opns $+$: `Date`, `Nat` $\to$ `Date`
...
attr part **opns** `matrNo-value`: `matrNo` $\to$ `Nat`
...
end

Fig. 2. Derived transition signature

e.g. `student`: `Enroll` $\to$ `Student` and `university`: `Enroll` $\to$ `University`. Attributes are connected to classes, e.g. `matrNo-class`: `matrNo` $\to$ `Student`, and to values, e.g. `matrNo-value`: `matrNo` $\to$ `Nat`. Sort `matrNo` represents a kind of container for attribute values and belongs to the graph part. The same holds for the association of this container to the corresponding class. The value of an attribute on the other hand is an attribute assignment and belongs to Ω^{attr}.

The interpretation of attributes as sorts leads to set–valued attributes. The restriction to single–valued total attributes can be realized by the following formula:

F1
s: `Student`, n, m: `matrNo`
always[
$\forall(n)\forall(m)\texttt{matrNo-class}(n) = \texttt{matrNo-class}(m) \Rightarrow n = m$ **(1)**
$\wedge$
$\forall(s)\exists(n)\texttt{matrNo-class}(n) = s$ **(2)**
]

If attributes in the graphical notation are regarded as single–valued and total, formula F1 has to be introduced for each attribute. Single–valued optional attributes are realized by (1) in F1. This would be the interpretation of attributes `lastDay`, `day`, and `passed` in Fig. 1. Considering the algebras wrt. transition signatures we obtain a natural notion of state and a suitable subset of the homomorphisms between these states can be interpreted as state transitions. This leads to the following notion of transition category.

Definition 12 Transition Category. The transition category

$$\textsc{Trans}(\Sigma) \subseteq \textsc{Alg}^p(\Sigma)$$

wrt. a transition signature $\Sigma = (\Sigma^G, \Sigma^D, \Omega^{attr})$ is given by the class of all objects of $\textsc{Alg}^p(\Sigma)$ and all morphisms in $\textsc{Alg}^p(\Sigma)$ whose Σ^D components are total.

Since transition signatures can be regarded as special signatures and transition categories are restrictions of algebra categories, our temporal logic defined in the previous section is directly applicable.

It remains to adapt the notion of sequential runs (see Def. 4) such that the data part is untouched during state transitions.

Definition 13 (Data Preserving Sequential Run). A data preserving sequential run $\underline{A}$ is a sequential run $\underline{A}: I \to \textsc{Trans}(\Sigma)$ such that $\underline{A}(i)|_{\Sigma^D} = D$ for all $i \in I$, where D is a fixed Σ^D algebra and $\underline{A}(i \to j)|_{\Sigma^D} = 1_D$ for all $i \to j \in I$.[13]

Definition 14 (Specification, Model). A specification for an information system

$$\textsc{Spec} = (\Sigma, \textsc{Req})$$

consists of a transition signature Σ, specifying the data structures to be stored in the information system and a set of closed temporal formulas $\textsc{Req}$ wrt. Σ, constraining the possibilities for state transitions.

Let $Z_0 \in \textsc{Trans}(\Sigma)$ be a state that satisfies all static integrity constraints implied by $\textsc{Req}$. The model class

$$\textsc{Mod}(\textsc{Spec}, Z_0)$$

for a specification $\textsc{Spec}$ and the initial state Z_0 is given by all data preserving sequential runs[14]in $\textsc{Trans}(\Sigma)$ starting at Z_0 and satisfying $\textsc{Req}$.

Now, our scenario described in Sect. 2 can be expressed as a formula wrt. transition signature $\textsc{Is}$.

F2

u: `University`, ex: `examination`, ld: `lastDay`, reg: `Registered`,
d: `day`, p: `passed`, sd: `systemDate`

always[
  $\forall(u)\forall(ex)\forall(reg)\forall(sd)($
    `university`$(reg) = u \wedge$ `examination`$(reg) = ex \wedge$
    `systemDate-class`$(sd) = u$
    $\Rightarrow$

[13] $|_{\Sigma^D}$ denotes the forgetful functor wrt. $\Sigma^D \subseteq \Sigma$.

[14] We assume that all sequential runs are based on the same infinite order I (compare Def. 4)

$\exists(ld)\texttt{lastDay-class}(ld) = ex \wedge$
$\texttt{systemDate-value}(sd) \leq \texttt{lastDay-value}(ld) + 1$
$\otimes$
$\exists(d)\exists(p)\texttt{day-class}(d) = ex \wedge \texttt{passed-class}(p) = ex$
)
]

This formula expresses the fact that either attribute `lastDay` of an examination is defined and the system date exceeds the value of this attribute at most by one day or attributes `day` and `passed` are defined ($\otimes$ denotes exclusive or). The information system construction in Sect. 5 motivates the $+1$ in $\texttt{systemDate-value}(sd) \leq \texttt{lastDay-value}(ld)+1$. Examinations can only be sat if $\texttt{systemDate-value}(sd) \leq \texttt{lastDay-value}(ld)$ (see rule R2). But they must be allowed to stay within the database also after their schedule is overrun in order to be handled in an appropriate way (see rule R4). We have arbitrarily decided that they must be processed one day after schedule overrun.

Note that our formulas are quite unhandy due to direct reliance on Def. 1 and the generated transition signature. These inconveniences should be removed by the introduction of suitable shorthands in a specification language on top of our formalism.

5 Construction

While temporal specifications (Sect. 4) axiomatically characterize well-behaving sequential runs in $\textsc{Trans}(\Sigma)$, a construction of an information system will provide an effective way to get all possible state transitions in an information system. In order to be independent of any concrete retrieval language and of the typical procedural aspects of these languages, we apply a rule based approach where a construction is given by a rewrite system.[15] The general idea is to specify a set of prototype transitions $r: L \to R$ as partial homomorphisms in $\textsc{Trans}(\Sigma)$ and generate the rewrite system by substitution and sequential composition. The operational effect specified by a prototype transition $r: L \to R$ will be the deletion of all objects in L not in the domain of r and the creation of all objects in R not in the image of r.

As has been shown in [19] substitution (i.e. pushout constructions) exists in $\textsc{Alg}^p(\Sigma)$ if and only if all operations in Σ are unary. This is the reason why our notion of transition signature (Def. 11) contains only unary operations in Σ^G. Due to our separation of objects and values, Σ^D may contain arbitrary operations. It has been shown in [21] that the transition category $\textsc{Trans}(\Sigma)$ is finitely co–complete. Therefore we get a substitution mechanism for a rule based definition of information system construction.

[15] This is an analogy to petri nets, where a net can be seen as a rewrite system induced by the rewriting rules of the contained transitions (reachability graph).

Definition 15 (Information System Construction). An information system construction

$$\textsc{Con} = (\Sigma, P)$$

consist of a transition signature Σ specifying the data structures of the system and a set P of transition rules, i.e. a set of morphisms in $\textsc{Trans}(\Sigma)$.

Each information system construction $\textsc{Con} = (\Sigma, P)$ induces a subcategory of $\textsc{Trans}(\Sigma)$, i.e. all transitions which are given by application of rules in P where application is defined by pushout constructions in $\textsc{Trans}(\Sigma)$.

Definition 16 (Rewrite Category). The rewrite category

$$\textsc{Rew}(\textsc{Con})$$

associated to an information system construction $\textsc{Con} = (\Sigma, P)$ is the subcategory of $\textsc{Trans}(\Sigma)$ which is generated by substitutions of P, i.e. $\textsc{Rew}(\textsc{Con}) = \langle \mathrm{Subst}(P) \rangle_{\textsc{Trans}(\Sigma)}$. The closure of P wrt. substitution in $\textsc{Trans}(\Sigma)$ is given by the set of arrows $\mathrm{Subst}(P) = \{r^\star : G \to H\}$ such that the following diagram is a pushout of a rule $(r: L \to R) \in P$ and an occurrence$(m: L \to G)$ in $\textsc{Trans}(\Sigma)$ where m is a total homomorphism.

$$\begin{tikzcd}
L \arrow[r, "r"] \arrow[d, "m"'] \arrow[dr, phantom, "\text{(PO)}"] & R \arrow[d, "m^\star"] \\
G \arrow[r, "r^\star"'] & H
\end{tikzcd}$$

As for specifications, we can define the semantics of a system construction in terms of data preserving sequential runs.

Definition 17 (Sequential Semantics of Construction). The sequential semantics

$$\textsc{Seq}(\textsc{Con}, Z_0)$$

of an information system construction $\textsc{Con} = (\Sigma, P)$ and an initial state $Z_0 \in \textsc{Trans}(\Sigma)$ is given by the set of all data preserving sequential runs $\underline{A}: I \to \textsc{Rew}(\textsc{Con})$ starting at Z_0 such that $\underline{A}(i \to i+1) \in \mathrm{Subst}(P)$ or $\underline{A}(i \to i+1) = 1_{\underline{A}(i)}$ for all $i \in I$, where $i+1$ is the least element of $\{j \in I \mid i \to j, i \neq j\}$.

The semantics is constructive, since each transition step of sequential runs in $\textsc{Rew}(\textsc{Con})$ corresponds to a rule application or no action. It is called sequential since rules are applied sequentially. (We also can define, e.g. parallel semantics where $\underline{A}(i \to i+1)$ is given by simultaneous application of more than one rule.)

Before discussing the relation of specification and construction of information systems in the following section, the rest of this section provides a construction for our running example by introducing suitable transition rules.

The first one describes the registration of an examination. Similar to the

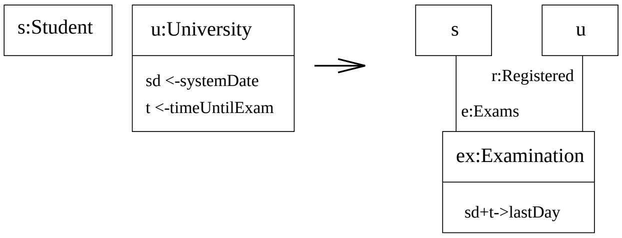

Fig. 3. Rule R1: Register examination

formal interpretation of schemes as transition signatures, the graphical notation used in rules like in Fig. 3 can canonically be transformed into a partial homomorphism in TRANS(Σ). We will only sketch this transformation.

The algebra on the left-hand side of rule R1 contains one element of sort `Student`, `University`, `systemDate`, and `timeUntilExam`, where the latter two are connected with the given university. The data part of this algebra is given by the term algebra $T_{\Sigma^D}(X)$ where X contains variable sd of sort `Date` and t of sort `Nat`.[16]

The right–hand side consists of the same student (indicated by the same name s), the same university (u), a new element of sort `Examination` and a new element of sort `lastDay`. Moreover new elements (i.e. relations) of sort `Exams` and `Registered` are introduced with appropriate connections. The value associated to the attribute `lastDay` is given by adding t days to date sd (indicated by $\rightarrow$, write access). The attributes `systemDate` and `timeUntilExam` remain the same (indicated by $\leftarrow$, read access).

The partial morphism is given in the obvious way: Nothing is deleted, and all elements are mapped onto their counterparts with the same name.

Fig. 4 describes a successful sitting of an examination within the given time limit. Since the university is only used to read the system date, $u \leftarrow$ `University`

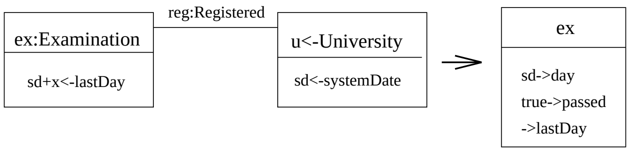

Fig. 4. Rule R2: Sit examination (successful)

indicates read access for this object. Although u does not appear on the right–

[16] Since we want to be able to express variable bindings and formal computations the data part Σ_D of rules is usually given by term algebras.

hand side, it is nevertheless preserved by the rule. The same holds for the relation *reg*. In the right–hand side $\rightarrow$ `lastDay` indicates that this attribute is to be removed. The given rule is only applicable if $sd + x$ (where x is a free variable of sort `Nat`) matches with the value of `lastDay` of examination ex. This means that there must be a solution of the equation $sd+x =$ `lastDay-value`(ex) which only exists if the system date is not beyond the last possible day of an examination.

Fig. 5 describes the deregistration of an examination. A deregistration is

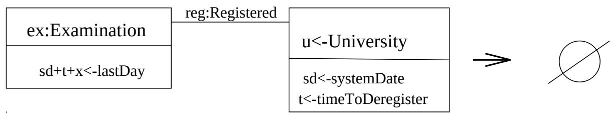

Fig. 5. Rule R3: Deregister examination

only possible if there are at least t days (given by attribute `timeToDeregister`) until the last possible day of the examination, i.e. the equation $sd + t + x =$ `lastDay-value`(ex) must have a solution. Since ex is not only used for read access (indicated by the colon in ex : `Examination`) and it does not appear on the right–hand side, it will be removed. Attribute `lastDay` will also be removed, due to the operational behavior of the pushout construction.

Fig. 6 deals with the case that the examination takes not place within the given time limit. The examination is regarded as sat unsuccessfully.

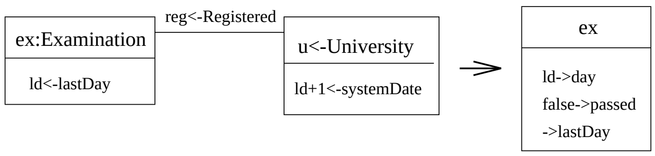

Fig. 6. Rule R4: Schedule overrun of examination

6 Correctness

After having introduced specification and construction of information systems in Sect. 4 and 5, respectively, the question of their relation will be discussed in this section. We present some ideas about suitable correctness notions between specification and construction. They are intended to be a starting point for further research on a practical methodology for information system development based on our approach.

A natural notion of correctness should require that the sequential runs given by the construction of an information system satisfy the requirements given by the specification. Technically, this could be expressed by $\textsc{Seq}(\textsc{Con}, Z) \subseteq \textsc{Mod}(\textsc{Spec}, Z)$ for all $Z \in \textsc{Trans}(\Sigma)$ satisfying the static integrity constraints implied by $\textsc{Spec}$. This notion is very strong since every admissible initial state must be considered. Often, one is only interested in certain aspects of the construction. This depends on what is regarded as especially important. To illustrate this point, let us turn to our running example. In our opinion the handling of pending examinations, i.e. registered examinations whose schedule is not overrun, is an important aspect.

To support a notion of correctness relative to certain aspects, we introduce a set $\mathcal{Z} \subseteq \textsc{Trans}(\Sigma)$ of initial states of interest.

Definition 18 (Correctness, first version). An information system construction $\textsc{Con} = (\Sigma, P)$ is correct wrt. a specification $\textsc{Spec} = (\Sigma, \textsc{Req})$[17] relative to a set of initial states of interest $\mathcal{Z} \subseteq \textsc{Trans}(\Sigma)$ if $\textsc{Seq}(\textsc{Con}, Z) \subseteq \textsc{Mod}(\textsc{Spec}, Z)$ for all $Z \in \mathcal{Z}$

Note that the strong notion of correctness is a special case of Def. 18 by taking $\mathcal{Z}$ as the set of all admissible initial states.

The set of states $\mathcal{Z}$ can be given either explicitly or specified by first order formulas. Considering our running example, the set $\mathcal{Z}$ should contain all states that satisfy static integrity constraints like formulas F1 and F2 (see Sect. 4) and should contain at least one pending examination whose schedule is not overrun, i.e., additionally the following formula should be satisfied:

F3

u: `University`, ex: `Examination`, ld: `lastDay`,
reg: `Registered`, sd: `systemDate`
$\exists(u)\exists(ex)\exists(ld)\exists(reg)\exists(sd)($
$\quad \texttt{university}(reg) = u \wedge \texttt{examination}(reg) = ex \wedge$
$\quad \texttt{systemDate-class}(sd) = u \wedge \texttt{lastDay-class}(ld) = ex \wedge$
$\quad \texttt{systemDate-value}(sd) \leq \texttt{lastDay-value}(ld)$
$\quad)$

Now we can ask the question if $\textsc{IsCon} = (\textsc{Is}, \{\mathrm{R1}, \ldots, \mathrm{R4}\})$ is correct wrt. $\textsc{IsSpec} = (\textsc{Is}, \textsc{Dax} \cup \{\mathrm{F2}\})$ relative to $\mathcal{Z}$ as defined above. Here $\textsc{Is}$ is the transition signature given in Fig. 2, R1, ..., R4 are the rules introduced in the previous section, $\textsc{Dax}$ contains all derived axioms describing totality/partiality of single–valued attributes, and F2 as given in Sect. 4.

A short look at our rule system reveals that there is no rule to increase the system date. Therefore our construction is correct in a trivial way. The problem is that our specification does not require any increase of the system date. Let us remedy this situation by introducing the following formula.

[17] Note that $\textsc{Con}$ and $\textsc{Spec}$ rely on the same transition signature Σ. The case $\textsc{Con} = (\Sigma', P)$ with $\Sigma \subseteq \Sigma'$ is an interesting extension to provide additional infrastructure in the construction but will not be regarded in this paper.

F4
always[
u: `University`, sd, sd': `systemDate`, d: `Date`
$\forall(u)\forall(sd)\forall(d)\exists(sd')($
`systemDate-class`$(sd) = u \wedge$ `systemDate-value`$(sd) = d$
$\Rightarrow$
sometime [`systemDate-class`$(sd') = u \wedge$ `systemDate-value`$(sd') > d$]
)
]

Obviously, our construction is not correct wrt. formula F4 since the system date cannot be increased So let us add rule R5 (Fig. 7) to IsCon. Unfortunately,

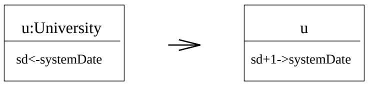

Fig. 7. Rule R5: Increase system date

our construction is still incorrect since we can construct a sequential run that in each step registers a new examination and never increases the system date. The problem is that rule R5 is applicable in each step at the same occurrence (at attribute `systemDate`) but never applied. To deal with this case we have to forbid infinite delays of certain rules (in this case R5). To get a sensible notion of infinite delay we first have to make some technical considerations.

Definition 19 (Strict Infinite Applicability). Let $\underline{A}: I \to \mathrm{Trans}(\Sigma)$ be a sequential run, $\mathrm{Con} = (\Sigma, P)$ an information system construction, and $r = (L \to R) \in P$ a rule. Then r is strictly infinitely applicable in $\underline{A}$ if there is an $i \in I$ and an occurrence $m: L \to \underline{A}(i)$ such that for all $i \to j \in I$ there exists an occurrence $m': L \to \underline{A}(j)$ with $I(i \to j) \circ m = m'$.

$$\begin{tikzcd} L \arrow[d, "m"] \arrow[drr, "m'"] & & \\ \underline{A}(i) \arrow[r] & \cdots \arrow[r] & \underline{A}(j) \arrow[r] & \cdots \end{tikzcd}$$

The adjective *strict* means infinite applicability at the *same* occurrence in contrast to infinite applicability at different occurrences. To make this distinction clear consider rule R5. We can easily construct a sequence $\underline{A}$ where in each step this rule has been applied, i.e. R5 has been applied infinitely. Nevertheless, since it relies on write access to attribute `systemDate`, technically it has been applied at different occurrences (this is an implication of the underlying graph transformation theory) and we therefore have no strict infinite applicability. Rules like R5 can be characterized as being destructive since it deletes items.

Let r be a transition rule being strictly infinitely applicable in a sequential run $\underline{A}$. Then there is an index $i \in I$ such that the conditions of Def. 19 hold. Two cases are possible:

1. Rule r has never been applied after index i. This means the initial occurrence $m: L \to \underline{A}(i)$ is preserved in $\underline{A}$ after index i and r has has been infinitely delayed at the same occurrence.
2. Rule r has been applied after index i. Then it is not destructive, i.e. it preserves the given occurrence.

Summarizing, we observe that strict infinite applicability leads to a notion of infinite delay only in connection with destructive rules. This motivates the following definition.

Definition 20 (Strict Infinite Delay Constraint). A strict infinite delay constraint (wrt. a transition signature Σ) is given by a proper partial morphism $d: L \to R \in \text{TRANS}(\Sigma)$, i.e. by a destructive rule.

A data preserving sequential run $\underline{A}: I \to \text{TRANS}(\Sigma)$ satisfies d if whenever there is an $i \in I$ and an occurrence $m: L \to \underline{A}(i)$ then d is not strictly infinitely applicable in $\underline{A}$.

This means that if rule d is applicable somewhere in $\underline{A}$ then it is not allowed to be infinitely delayed at the same occurrence. But it can be infinitely delayed wrt. to changing occurrences. To prohibit the latter a different notion of delay constraint is necessary.[18]

Let $\mathcal{D}$ be a set of strict infinite delay constraints and $\text{SEQ}(\text{CON}, Z_0, \mathcal{D})$ contain all sequential runs $\underline{A} \in \text{SEQ}(\text{CON}, Z_0)$ that satisfy all $d \in \mathcal{D}$.

Definition 21 (Correctness, second version). An information system construction $\text{CON} = (\Sigma, P)$ is correct wrt. a specification $\text{SPEC} = (\Sigma, \text{REQ})$ and a set of strict infinite delay constraints $\mathcal{D}$ relative to $\mathcal{Z} \subseteq \text{TRANS}(\Sigma)$ if $\text{SEQ}(\text{CON}, Z, \mathcal{D}) \subseteq \text{MOD}(\text{SPEC}, Z)$ for all $Z \in \mathcal{Z}$

Taking $\mathcal{D} = \{\text{R5}\}$ we actually get correctness of IsCON wrt. formula F4.

Let us now resume the correctness considerations wrt. formula F2. Since rule R5 is always applicable we can construct a sequential run such that the system date exceeds the last possible day of a pending examination by two days. This means that IsCON is not correct wrt. F2. The reason for this is that rule R4 is not applied one day after schedule overrun of the examination but rule R5. Technically, R5 is in conflict with R4 as made precise in the following definition.

Definition 22 (Conflict). A rule $r' = (L' \to R')$ is in conflict with a rule $r = (L \to R)$ at occurrences $m': L' \to A$ and $m: L \to A$, where $A \in \text{TRANS}(\Sigma)$, if $r'^{\star} \circ m$ is not total. Here $r'^{\star}$ is given by the pushout resulting from application of rule r' at occurrence m'.

[18] We could call this an *infinite delay constraint.*

$$\begin{tikzcd}
 & L \arrow[r, "r"] \arrow[d, "m"] & R \\
L' \arrow[r, "m'"] \arrow[d, "r'"'] \arrow[dr, phantom, "(\mathrm{PO})"] & A \arrow[d, "r'^{*}"] & \\
R' \arrow[r, "m'^{*}"] & A' &
\end{tikzcd}$$

Intuitively, r' is in conflict with r at m' and m if the application of r' at m' removes elements used by the occurrence m for r. In our example, attribute `systemDate` is removed and replaced by a new one where the date value is incremented by one day. To provide a strategy for rule application in case of conflicts, we introduce a *domination relation* $\rhd$ between rules.

Definition 23 (Conflict Resolution). Let $\underline{A}: I \to \mathrm{T{\scriptsize RANS}}(\Sigma)$ be a sequential run, $\mathrm{C{\scriptsize ON}} = (\Sigma, P)$ an information system construction, and $r, r' \in P$ be two rules. $\underline{A}$ satisfies $r \rhd r'$ if whenever r and r' are in conflict at $\underline{A}(i), i \in I$, then r is applied (r dominates r').

Let $\mathrm{S{\scriptsize EQ}}(\mathrm{C{\scriptsize ON}}, Z_0, \mathcal{D}, \rhd)$ contain all sequential runs $\underline{A} \in \mathrm{S{\scriptsize EQ}}(\mathrm{C{\scriptsize ON}}, Z_0, \mathcal{D})$ that satisfy $\rhd$.

Definition 24 (Correctness, third version). An information system construction $\mathrm{C{\scriptsize ON}} = (\Sigma, P)$ is correct wrt. a specification $\mathrm{S{\scriptsize PEC}} = (\Sigma, \mathrm{R{\scriptsize EQ}})$, a set of infinite delay constraints $\mathcal{D}$, and a domination relation $\rhd$ relative to $\mathcal{Z} \subseteq \mathrm{T{\scriptsize RANS}}(\Sigma)$ if $\mathrm{S{\scriptsize EQ}}(\mathrm{C{\scriptsize ON}}, Z, \mathcal{D}, \rhd) \subseteq \mathrm{M{\scriptsize OD}}(\mathrm{S{\scriptsize PEC}}, Z)$ for all $Z \in \mathcal{Z}$

Taking $\rhd = \{(\mathrm{R4}, \mathrm{R5})\}$ as the domination relation in our running example we get also correctness wrt. F2. Although the third version is suitable for our running example, it has to be validated by further examples.

7 Conclusion

In this paper, we have presented an algebraic formulation of a temporal logic suitable for the description of dynamic properties in the conceptual modeling of information systems. Static properties were expressed also in an algebraic manner by a corresponding interpretation of signatures. Thereby, we have integrated both static and dynamic aspects in a formal way. Based on this solid algebraic fundament we were able to describe information systems in a descriptive and in an operational way by means of specifications and constructions, respectively. A correctness criterion established a link between specifications and constructions.

For sake of shortness we only have introduced one elementary temporal quantifier (**sometime**) in our temporal logic in Sect. 3. Additional quantifiers can be defined in the same way.

To discuss a running example we introduced a graphical notation for schemes and rules and sketched its formal interpretation based on the semantical domain $\text{ALG}^p(\Sigma)$. The temporal logic has been used to specify static and dynamic integrity constraints that capture the scenary informally discussed in Sect. 2. The notations have been chosen in an ad hoc manner and are only intended for illustrative purpose. As has already been stated in Sect. 4 the logic is not intended to be used directly for specification. Therefore the development of a suitable specification and construction formalism with graphical notations for schemes and rules and a specification language based on temporal formulas is a necessary prerequisite to apply our ideas on larger examples.

Accordingly the correctness relation has to be validated in practically relevant situations. We assume that our notion of conflict resolution (Def. 23) is too coarse to handle all situations in the right way.

Another question pertains to a proof method for correctness. In Sect. 6 we only have discussed the nature of our correctness notion and provided a tentative definition. Correctness in our example has only be discussed informally. A systematic procedure to assist in correctness proofs remains to be investigated.

Strongly connected with the proof method is the question of a development method. How does the development of a specification and a corresponding construction proceed? Are they done sequentially or intertwined? An important point here is the derivation of an actual implementation (e.g. for a specific database system) from our abstract constructions.

Other notions of semantics for constructions seem also be interesting. The introduction of parallel rules might introduce new operational effects that cannot be achieved by sequential application of rules (compare [19]). We can envisage a situation where parallel semantics is correct wrt. a given specification and sequential semantics is not. This case would introduce some kind of collaboration constraint.

The theoretical foundation presented in this paper is intended to specify and construct an information system at a given moment in time. But the important question of evolution of such a system due to user requirements or technological change has not been touched. Actually, we think the evolution of systems is an extremely important question that formal methods should try to tackle. Therefore we would like to combine our work on scheme evolution as given in [6] with specification as introduced in this paper. This would raise interesting questions like the validity of formulas after modification of the system or how a given construction has to be modified to satisfy modified formulas (requirements).

One perspective that has not been mentioned yet is that the presented theoretical foundation can also serve as a semantical background for specification and construction of Petri nets in the same spirit as in the work of Reisig [24]. Transforming Petri nets into graph structures as described in [20], we immediately can apply our temporal logic to specify properties of such nets. Interestingly, our notion of infinite delay (Def. 20) then seems to correspond to the notion of progress in [24].

References

1. S. Abiteboul and R. Hull. IFO – A Formal Semantic Database Model. *ACM Trans. on Database Systems*, 12(4):525–565, 1987.
2. A. Asperti and G. Longo. *Categories, Types and Structures.* Foundations of Computing Series. MIT Press, 1991.
3. C. Beeri. A Formal Approach to Object-Oriented Databases. *Data & Knowledge Engineering*, 5(4):353–382, 1990.
4. G. Booch. *Object Oriented Analysis and Design.* Benjamin/Cummings, 2nd edition, 1994.
5. P. P. Chen. The entity–relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
6. I. Claßen and M. Löwe. Scheme evolution in object–oriented models. In M. Wirsing, editor, *ICSE–17 Workshop on Formal Methods Application in Software Engineering Practice*, Seattle, Washington, April 1995.
7. I. Claßen, M. Löwe, S. Waßerroth, and J. Wortmann. Static and dynamic semantics of entity–relationship models based on algebraic methods. In B. Wolfinger, editor, *Innovationen bei Rechen– und Kommunikationssystemen*, Informatik aktuell, pages 2–9. Springer–Verlag, 1994.
8. I. Claßen, M. Löwe, S. Waßerroth, and J. Wortmann. Algebraic semantics for object–oriented modelling. Technical Report 95–10, Technische Universität Berlin, 1995.
9. H.-D. Ehrich, U. W. Lipeck, and M. Gogolla. Specification, Semantics, and Enforcement of Dynamic Database Constraints. In *Proc. Int. Conf. on Very Large Databases VLDB '84*, pages 301–308, Singapore, 1984.
10. R. Elmasri, J. Weeldreyer, and A. Hevner. The Category Concept: An Extension to the Entity-Relationship Model. *Data & Knowledge Engineering*, 1:75–116, 1985.
11. G. Engels, M. Gogolla, U. Hohenstein, K. Hülsmann, P. Löhr-Richter, G. Saake, and H.-D. Ehrich. Conceptual Modelling of Database Applications using an Extended ER Model. *Data & Knowledge Engineering, North-Holland*, 9(2):157–204, 1992.
12. M. Gogolla and R. Herzig. An Algebraic Semantics for the Object Specification Language TROLL *light*. In E. Astesiano, G. Reggio, and A. Tarlecki, editors, *Proc. 10th Workshop on Abstract Data Types (ADT'94)*, pages 288–304. Springer, Berlin, LNCS 906, 1995.
13. F. Golshani, T.S.E. Maibaum, and M.R. Sadler. A Modal System of Algebras for Database Specification and Query/Update Language. In *9th VLDB*, pages 331–339, 1983.
14. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.
15. R. Hull and R. King. Semantic Database Modelling: Survey, Applications, and Research Issues. *ACM Computing Surveys*, 19(3):201–260, 1987.
16. I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard. *Object–Oriented Software Engineering — A Use Case Driven Approach.* Addison–Wesley, 1993. Revised Fourth Printing.
17. U. W. Lipeck and G. Saake. Monitoring Dynamic Integrity Constraints Based on Temporal Logic. *Information Systems*, 12:255–269, 1987.
18. U. W. Lipek. *Dynamische Integrität von Datenbanken*, volume 209 of *Informatik Fachberichte.* Springer, 1989.

19. M. Löwe. Algebraic approach to single-pushout graph transformation. *Theoretical Computer Science*, 109:181 – 224, 1993. Short version of Techn. Rep. 90/05, TU Berlin, Department of Computer Science, 1990.
20. M. Löwe. Von Petri–Netzen zu Graphgrammatiken und zurück. In F. Simon, editor, *Proc. GI–Workshop "Deklarative Programmierung und Spezifikation, Bericht Nr. 9412"*, pages 79–82, Kiel, 1994. Christian–Albrechts–Universität Kiel. Extended Abstract.
21. M. Löwe, M. Korff, and A. Wagner. An algebraic framework for the transformation of attributed graphs. In M.R. Sleep, M.J. Plasmeijer, and M.C. van Eekelen, editors, *Term Graph Rewriting: Theory and Practice*, chapter 14, pages 185–199. John Wiley & Sons Ltd, 1993.
22. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer, 1992.
23. J. Peckham and F. Maryanski. Semantic Data Models. *ACM Computing Surveys*, 20(3):153–189, 1988.
24. W. Reisig. Progress in petri nets. Technical Report Informatik–Bericht Nr. 34, Humboldt–Universität zu Berlin, 1994.
25. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object–oriented Modeling and Design.* Prentice–Hall International Editions. Prentice–Hall, 1991.
26. M. Schrefl, A M. Tjoa, and R.R. Wagner. Comparison Criteria for Semantic Data Models. In C.V. Ramamoorthy, editor, *Proc. Int. Conf. on Data Engineering*, pages 120–125. IEEE, Silver Spring (MD), 1984.
27. S. D. Urban and L. Delcambre. An Analysis of the Structural, Dynamic, and Temporal Aspects of Semantic Data Models. In *Proc. Int. Conf. on Data Engineering*, pages 382–387. ACM, New York, 1986.