Declarative Spatial Reasoning in Temporal Here-and-There with Constraints

François Olivier[1], Martín Diéguez[2] and Carl Schultz[3]

[1]Sorbonne University, France
[2]University of Angers, France
[3]Aarhus University, Denmark

Abstract

Declarative Spatial Reasoning (DSR) uses analytic geometry to represent objects by parameters and spatial relations by equations and inequalities over those parameters. We define reusable spatial schemas in this spirit and show how, when paired with instance-specific components, they induce specialisations of Temporal Here-and-There with Constraints (THTc), a foundational logic for temporal Constraint Answer Set Programming. We call this framework THTc(DSR). Each particular THTc specialisation enables temporal and nonmonotonic reasoning over heterogeneous qualitative and quantitative spatial information, declarative inertia through a uniform axiom schema, temporal shifts at both formula and object-argument levels, and automatic spatial ramification. Stable models determine qualitative spatio-temporal scenarios together with numerical witnesses from which concrete diagrammatic representations can be constructed.

Keywords

Answer Set Programming, Temporal Here-and-There with Constraints, Declarative Spatial Reasoning, Qualitative Spatial Reasoning, spatial constraints, nonmonotonic reasoning, spatio-temporal reasoning

CEUR-WS.org/Vol-4311/ASPOCP_regular_12.pdf

1. Introduction

Declarative Spatial Reasoning (DSR) [1] is an approach that enables qualitative and quantitative spatial reasoning over heterogeneous objects. It provides a flexible alternative to widely used Qualitative Spatial Reasoning calculi [2], allowing relation definitions to be adapted to particular applications and different spatial aspects to coexist within the same representation. DSR relies on analytic geometry: objects are described by numerical parameters, while spatial relations are interpreted by equations and inequalities involving these parameters.

In order to enable nonmonotonic DSR, the framework has been integrated into several Constraint Answer Set Programming systems [3, 4, 5]. Among them, ASPMT(QS) [3] provides the most explicit logical account, building on the circumscription-based characterisation of stable models and supporting spatial defaults, action-dependent frame axioms, and indirect spatial effects. This account is nevertheless operationalised through completion-based transformations of suitably restricted theories into SMT, which limits its flexibility at the semantic level. More generally, these systems encode temporal evolution and actions explicitly rather than within a uniform temporal constraint semantics.

In this paper, we integrate DSR into Temporal Here-and-There with Constraints (THTc) [6], a foundational logic for temporal Constraint Answer Set Programming. We define reusable spatial schemas in the spirit of DSR and show how, when paired with instance-specific components, they induce specialisations of THTc. We call the resulting framework THTc(DSR). Besides the flexibility afforded by this fully logical foundation, each particular THTc specialisation supports temporal and nonmonotonic reasoning over heterogeneous qualitative and quantitative spatial information, declarative inertia through a uniform axiom schema, temporal shifts at both formula and object-argument levels, and automatic spatial ramification.

Joint Proceedings of the Workshops and Doctoral Consortium of the 42nd International Conference on Logic Programming (ICLP 2026), co-located with the Federated Logic Conference (FLoC 2026), July 18–19 and 24–25, 2026, Lisbon, Portugal.
francois.olivier@sorbonne-universite.fr (F. Olivier); martin.dieguezlodeiro@univ-angers.fr (M. Diéguez); cschultz@ece.au.dk (C. Schultz)
0009-0003-5810-8022 (F. Olivier); 0000-0003-3440-4348 (M. Diéguez); 0000-0001-7334-6617 (C. Schultz)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-03

To illustrate these features, we use throughout the paper the following *growing-circle example*:

1. Let a and b be two points, c a circle with radius 1.5, and d a rectangle of height 5.
2. Initially, point a lies to the right of circle c and to the left of rectangle d, and is horizontally aligned with the centre of circle c.
3. Point b is initially horizontally aligned with point a. At the next step, the position of point b is constrained to lie to the left of the initial position of point a.
4. At any time, a propositional condition p may become true, triggering an increase in the radius of circle c until external connection with rectangle d is reached.
5. The remaining object parameters persist by default.

The example combines temporal evolution, numeric constraints, nonmonotonic defaults, and spatial ramification. In particular, the containment of a follows from the effect of $grows_{\mathsf{c}}(c)$, rather than being stated as an additional effect, while unchanged geometric parameters persist through inertia.

The paper is structured as follows. Section 2 introduces spatial domains, which specify the reusable spatial and action semantics. Section 3 explains how a spatial domain induces a THTc signature and denotations. Section 4 presents reusable modelling axioms and develops the growing-circle example through scenarios and diagrams. Section 5 discusses related work, scope, and future directions.

2. Spatial Domains

A *spatial domain*

$$\mathcal{S} = (\mathfrak{S}_{\mathcal{L}}, \mathcal{I})$$

specifies the spatial vocabulary and constraints used to interpret spatial atoms independently of any particular logical or computational implementation. Its generic *schema structure* $\mathfrak{S}_{\mathcal{L}}$ is expressed in a schema language based on the mathematical language $\mathcal{L}$. It fixes the geometric sorts, parameter and constraint schemas, action symbols, and action-effect schemas in terms of schematic object variables. The *instance* $\mathcal{I}$, by contrast, supplies the concrete object names, ambient-space bounds, and finite trace length for a particular case. Its objects and bounds instantiate the corresponding schemas. Since $\mathfrak{S}_{\mathcal{L}}$ is independent of any particular instance, it can be reused across different instances.

Formally,

$$\mathfrak{S}_{\mathcal{L}} = \langle \mathcal{G}, \mathrm{param}, \mathrm{derived}, \mathrm{degen}, \mathrm{lim}, \mathcal{R}, \mathrm{constr}, \mathcal{A}, \mathrm{effect} \rangle\,, \qquad \mathcal{I} = \langle \mathcal{O}, (I_x, \ldots, I_z), \lambda \rangle\,.$$

Here, $\mathcal{G}$, $\mathcal{R}$, and $\mathcal{A}$ are finite sets. The components param, $\mathrm{derived}$, degen, lim, and constr define schemas in $\mathcal{L}$, while effect defines action-effect schemas in the extension of $\mathcal{L}$ that permits one-step future shifts. The components $\mathcal{O}$, $(I_x, \ldots, I_z)$, and λ provide, respectively, the concrete object names, ambient-space bounds, and finite trace length of the instance.

For illustration, throughout this paper we use a schema structure based on QF_NRA, the language of quantifier-free nonlinear real arithmetic, whose formulas are Boolean combinations of polynomial equalities and inequalities over $\mathbb{R}$. We now describe each component.

2.1. Geometric Sort Schemas

Geometric sorts and object constant names. The component $\mathcal{G}$ from $\mathfrak{S}_{\mathcal{L}}$ is a finite set of *geometric sorts*. In the running example, we use

$$\mathcal{G} = \{\mathsf{point}, \mathsf{circle}, \mathsf{rect}\}.$$

Other sorts, such as segment, triangle, or polygon, could be included provided that their geometry can be described by constraints in $\mathcal{L}$. These sorts determine the kinds of spatial objects that may occur in instances. We often abbreviate geometric sorts by their initial letter (e.g. p for point, c for circle, r for rectangle, etc.).

Set $\mathcal{O}$ from instance $\mathcal{I}$ is a family

$$\mathcal{O} = \{\mathcal{O}_{\mathfrak{g}}\}_{\mathfrak{g}\in\mathcal{G}}$$

of finite, pairwise disjoint sorted sets, each containing *object constants* (or *object names*) that serve as syntactic identifiers of geometric objects. Accordingly, in the growing-circle example, set $\mathcal{O}$ contains the subsets

$$\mathcal{O}_{\mathsf{point}} = \{a, b\}, \quad \mathcal{O}_{\mathsf{circle}} = \{c\}, \quad \mathcal{O}_{\mathsf{rect}} = \{d\}.$$

We write *schematic object variables* with capital letters; they are sorted placeholders independent of any particular spatial instance. For the sorts point, circle, and rect, we use P, C, R, respectively, with indexed variants such as P_1, P_2 when several schematic object variables of the same sort are needed. When a specific instance $\mathcal{I}$ is considered, we write $o_{\mathfrak{g}}$ for a metavariable ranging over $\mathcal{O}_{\mathfrak{g}}$. The sort annotation may be omitted when it is recoverable from context, yielding $o, o_1, o_2, \ldots$.

Parameter schemas. The component param specifies for each geometric sort $\mathfrak{g} \in \mathcal{G}$ a finite set of *parameter schemas*, each obtained by attaching a parameter name to a schematic object variable of sort $\mathfrak{g}$. From an analytic-geometric perspective, parameters characterise an object in a Cartesian coordinate system. For each sort, some parameters serve as *anchor parameters*, fixing the object's absolute position in space, while the remaining *shape parameters* describe its geometry relative to this anchor, such as its size or extent. Examples of parameter schemas are:

$\mathrm{param}(\mathsf{point}) \stackrel{def}{=} \{x_P, y_P\}$ (Cartesian anchor coordinates of a point),
$\mathrm{param}(\mathsf{circle}) \stackrel{def}{=} \{x_C, y_C, \mathit{radius}_C\}$ (centre anchor and radius of a circle),
$\mathrm{param}(\mathsf{rect}) \stackrel{def}{=} \{\mathit{leftSide}_R, \mathit{bottom}_R, \mathit{width}_R, \mathit{height}_R\}$ (bottom-left anchor of an axis-aligned rectangle, width, and height).

Parameter schemas constitute the primitive atomic $\mathcal{L}$-term schemas of the schema language, from which the remaining schemas of the structure are constructed.

Derived-quantity schemas. The component $\mathrm{derived}$ assigns to each tuple of sorts $(\mathfrak{g}_1, \ldots, \mathfrak{g}_n)$ a possibly empty finite mapping from *derived-quantity schemas*, formed with schematic object variables of those sorts, to $\mathcal{L}$-term schemas constructed from the corresponding parameter schemas using the term-forming operations of $\mathcal{L}$. Each derived-quantity schema is an abbreviation for its associated $\mathcal{L}$-term schema and, although non-primitive, is treated as an atomic $\mathcal{L}$-term schema in subsequent constructions.

We distinguish between *intrinsic* derived-quantity schemas, for which $n = 1$ and which depend on a single object, and *multi-object* derived-quantity schemas, for which $n > 1$ and which depend on several objects. For instance, for axis-aligned rectangles, we use the intrinsic derived-quantity schemas

$$\mathrm{derived}(\mathsf{rect}) \stackrel{def}{=} \left\{ \begin{array}{rcl} \mathit{rightSide}_R & \mapsto & \mathit{leftSide}_R + \mathit{width}_R, \\ \mathit{top}_R & \mapsto & \mathit{bottom}_R + \mathit{height}_R \end{array} \right\}.$$

A typical multi-object derived-quantity schema is the squared Euclidean distance between two points:

$$\mathrm{derived}(\mathsf{point}, \mathsf{point}) \stackrel{def}{=} \left\{ \mathit{sqDist}_{P_1,P_2} \mapsto (x_{P_1} - x_{P_2})^2 + (y_{P_1} - y_{P_2})^2 \right\}.$$

Squared distance is sufficient for distance comparisons and avoids introducing square roots into the constraint language.

For each tuple of sorts $(\mathfrak{g}_1, \ldots, \mathfrak{g}_n)$, let $\mathrm{term}_{\mathcal{L}}(\mathfrak{g}_1, \ldots, \mathfrak{g}_n)$ be the smallest set of $\mathcal{L}$-term schemas that contains the parameter schemas in $\mathrm{param}(\mathfrak{g}_1) \cup \cdots \cup \mathrm{param}(\mathfrak{g}_n)$, contains all derived-quantity schemas defined for $(\mathfrak{g}_1, \ldots, \mathfrak{g}_n)$ or any of its subtuples, and is closed under the term-forming operations of $\mathcal{L}$. For example, $\mathrm{term}_{\mathcal{L}}(\mathsf{point}, \mathsf{rect})$ contains the parameter schemas for points and rectangles, $\mathit{rightSide}_R$, top_R, and all compound $\mathcal{L}$-term schemas formed from them. The $\mathcal{L}$-terms of the schema language are simply all such schemas, without restriction to any particular sorts.

Degeneracy formula schemas. The component degen assigns to each geometric sort $\mathfrak{g} \in \mathcal{G}$ an $\mathcal{L}$-formula schema over $\mathrm{term}_{\mathcal{L}}(\mathfrak{g})$ that characterises when an object of that sort is degenerate.

For instance, for the circle sort, we define

$$\mathrm{degen}(\mathsf{circle}) \stackrel{def}{=} \mathit{radius}_C \leq 0,$$

so that a circle is degenerate exactly when its radius is not strictly positive.

Limit schemas and space bounds. The component lim assigns to each geometric sort $\mathfrak{g} \in \mathcal{G}$ an $\mathcal{L}$-formula schema over $\mathrm{term}_{\mathcal{L}}(\mathfrak{g})$, written using the ambient-space bounds $(I_x, \ldots, I_z)$ supplied by the instance and characterising when an object violates them. In two dimensions, we write (I_x, I_y); in three dimensions, (I_x, I_y, I_z). Each interval is written

$$I_x = [x_{\min}, x_{\max}], \qquad I_y = [y_{\min}, y_{\max}], \qquad I_z = [z_{\min}, z_{\max}],$$

when the corresponding coordinate is present. For example,

$$I_x = [0, 10], \qquad I_y = [0, 10]$$

defines a bounded two-dimensional space. Using these bounds, we define an $\mathcal{L}$-formula schema characterising bound violations. For points, for instance,

$$\mathrm{lim}(\mathsf{point}) \stackrel{def}{=} x_P < x_{\min} \ \vee \ x_P > x_{\max} \ \vee \ y_P < y_{\min} \ \vee \ y_P > y_{\max}.$$

An unbounded direction is represented by omitting the corresponding lower or upper bound. We then write $\pm\infty$ metalinguistically for this omission.

2.2. Relation schemas

Spatial relation symbols. The component $\mathcal{R}$ is a finite set of *typed spatial relation symbols*, partitioned into positive relation symbols $\mathcal{R}^+$ and their negative counterparts $\mathcal{R}^-$. A positive relation symbol has the form

$$r_{\mathfrak{g}_1 \ldots \mathfrak{g}_n},$$

where r is the relation name and $(\mathfrak{g}_1, \ldots, \mathfrak{g}_n)$ is its type, with each $\mathfrak{g}_i \in \mathcal{G}$. For example, $\mathit{left}_{\mathsf{pp}}$ relates two points, while $\mathit{inside}_{\mathsf{pr}}$ relates a point and a rectangle.

For every $r_{\mathfrak{g}_1 \ldots \mathfrak{g}_n} \in \mathcal{R}^+$, the set $\mathcal{R}^-$ contains a negative counterpart, written $-r_{\mathfrak{g}_1 \ldots \mathfrak{g}_n}$. The set $\mathcal{R}^+$ may include directional, topological, metric, or domain-specific spatial relations, and is not restricted to a fixed qualitative calculus.

Relation-constraint schemas. The component constr assigns to each typed spatial relation symbol $r_{\mathfrak{g}_1 \ldots \mathfrak{g}_n}$ an $\mathcal{L}$-formula schema over $\mathrm{term}_{\mathcal{L}}(\mathfrak{g}_1, \ldots, \mathfrak{g}_n)$. For instance,

$$\mathrm{constr}(\mathit{left}_{\mathsf{pp}}) \stackrel{def}{=} x_{P_1} < x_{P_2}.$$

Negative counterparts are interpreted by mathematical negation of the corresponding positive schema. For example,

$$\mathrm{constr}(-\mathit{left}_{\mathsf{pp}}) \stackrel{def}{=} x_{P_1} \geq x_{P_2}.$$

A negative counterpart need not coincide with another positive relation. For instance, the relation $-\mathit{left}_{\mathsf{pp}}$ captures failure of leftness, but does not correspond to the positive relation $\mathit{right}_{\mathsf{pp}}$.

Other spatial relations are defined analogously, provided that they can be expressed in the language $\mathcal{L}$ using schemas from $\mathrm{term}_{\mathcal{L}}(\mathfrak{g}_1, \ldots, \mathfrak{g}_n)$. For example,

$$\mathrm{constr}(\mathit{inside}_{\mathsf{pc}}) \stackrel{def}{=} (x_P - x_C)^2 + (y_P - y_C)^2 < \mathit{radius}_C^2,$$

which already shows that $\mathcal{L}$ must include at least quadratic polynomial constraints to express containment relations of this kind.

2.3. Action schemas

Action symbols. The component $\mathcal{A}$ is a finite set of typed action symbols. An action symbol has the form

$$a_{\mathfrak{g}_1 \dots \mathfrak{g}_n}$$

where a is the action performed and $(\mathfrak{g}_1, \dots, \mathfrak{g}_n)$ is its type, with each $\mathfrak{g}_i \in \mathcal{G}$. For example, $grows_{\mathrm{c}}$ denotes the action of growing performed on a circle.

Action-effect schemas. The component effect assigns to each typed action symbol $a_{\mathfrak{g}_1 \dots \mathfrak{g}_n}$ an $\mathcal{L}$-formula schema over $\mathrm{term}_{\mathcal{L}}(\mathfrak{g}_1, \dots, \mathfrak{g}_n)$. Additionally in these schemas, some occurrences of atomic $\mathcal{L}$-term schemas are prefixed by $\bigcirc$, indicating that their instantiated values are evaluated at the next state rather than the current one. The schema describes how an occurrence of the action constrains its arguments from the current state to the next.

For instance, for the growing-circle example,

$$\mathrm{effect}(grows_{\mathrm{c}}) \stackrel{def}{=} \bigcirc radius_C > radius_C.$$

The schema describes one growth step; its triggering, repetition, and termination remain part of the scenario theory.

3. Integrating Spatial Domains in THTc

A general account of Temporal Here-and-There with Constraints (THTc) is given in [6]. We now explain how a schema structure $\mathfrak{S}_{\mathcal{L}}$, together with a spatial instance $\mathcal{I}$, induces the THTc signature and semantics for spatial constraints. In this construction, the schematic object variables used in $\mathfrak{S}_{\mathcal{L}}$ are not part of the THTc signature itself: they serve only to define schemas, which are instantiated by replacing them with concrete object names from $\mathcal{I}$.

3.1. Syntax

THTc is based on a constraint signature

$$(\mathcal{X}, \mathcal{D}, \mathcal{C}),$$

where $\mathcal{X}$ is a set of variables, $\mathcal{D}$ is a non-empty domain of values, and $\mathcal{C}$ is a set of constraint atoms. We describe each component in turn.

Variables. The variable set is

$$\mathcal{X} \stackrel{def}{=} \mathcal{X}_{\mathcal{P}} \cup \mathcal{X}_{\mathcal{A}} \cup \mathcal{X}_{\mathrm{param}},$$

where $\mathcal{X}_{\mathcal{P}}$ is a set of ordinary propositional variables, $\mathcal{X}_{\mathcal{A}}$ is the set of variables formed by applying typed action symbols to tuples of unshifted object arguments, and $\mathcal{X}_{\mathrm{param}}$ is obtained by instantiating the parameter schemas of $\mathfrak{S}_{\mathcal{L}}$ with object names from $\mathcal{I}$.

More precisely,

$$\mathcal{X}_{\mathcal{A}} \stackrel{def}{=} \{a_{\mathfrak{g}_1 \dots \mathfrak{g}_n}(o_1, \dots, o_n) \mid a_{\mathfrak{g}_1 \dots \mathfrak{g}_n} \in \mathcal{A},\ o_i \in \mathcal{O}_{\mathfrak{g}_i}\}.$$

For a schema with argument sorts $(\mathfrak{g}_1, \dots, \mathfrak{g}_n)$, we write $[o_1, \dots, o_n]$ for its instantiation by $o_i \in \mathcal{O}_{\mathfrak{g}_i}$. Thus,

$$\mathcal{X}_{\mathrm{param}} \stackrel{def}{=} \bigcup_{\mathfrak{g} \in \mathcal{G}} \bigcup_{o \in \mathcal{O}_{\mathfrak{g}}} \{x[o] \mid x \in \mathrm{param}(\mathfrak{g})\}.$$

We write $\mathcal{X}_{\mathrm{param}}|_o$ for the restriction of $\mathcal{X}_{\mathrm{param}}$ to the parameter variables associated with object o.

In the growing-circle example, $\mathcal{X}_{\mathcal{P}} = \{p\}$, $\mathcal{X}_{\mathcal{A}} = \{grows_{\mathrm{c}}(c)\}$, and

$$\mathcal{X}_{\mathrm{param}} = \{x_a, y_a, x_b, y_b, x_c, y_c, radius_c, leftSide_d, bottom_d, width_d, height_d\}.$$

Instantiated derived-quantities are abbreviations over these variables and are eliminated by expansion.

Value domain. The value domain is

$$\mathcal{D} = \{\mathfrak{t}\} \cup \mathbb{D},$$

where $\mathfrak{t}$ stands for "true" and $\mathbb{D}$ is the numerical domain associated with $\mathcal{L}$. Here, $\mathbb{D} = \mathbb{R}$, since $\mathcal{L} = \text{QF_NRA}$. Instance bounds are imposed separately by background axioms introduced later.

Constraint atoms. The set of constraint atoms is

$$\mathcal{C} \stackrel{def}{=} \mathcal{C}_{\mathcal{P}} \cup \mathcal{C}_{\mathcal{A}} \cup \mathcal{C}_{\mathcal{N}} \cup \mathcal{C}_{\mathcal{R}}.$$

Sets $\mathcal{C}_{\mathcal{P}}$ and $\mathcal{C}_{\mathcal{A}}$ contain the regular atoms corresponding, respectively, to variables in $\mathcal{X}_{\mathcal{P}}$ and $\mathcal{X}_{\mathcal{A}}$. Set $\mathcal{C}_{\mathcal{N}}$ consists of numerical atoms

$$\tau_1 \bowtie \tau_2,$$

where $\bowtie \in \{=, \neq, <, \leq, >, \geq\}$ and τ_1, τ_2 are $\mathcal{L}$-terms built from numerical constants and temporally shifted instances of atomic term schemas, with instances of derived-quantity schemas understood through their expansions under $\mathrm{derived}$. We write $\bigcirc^{\ell} x$ for the value of a parameter variable $x \in \mathcal{X}_{\text{param}}$ shifted by $\ell \in \mathbb{Z}$ time steps, identifying x with $\bigcirc^0 x$, and abbreviating $\bigcirc^1 x$ as $\bigcirc x$ and $\bigcirc^{-i} x$ as $\bullet^i x$.

Numerical atoms explicitly included in a scenario theory are restricted to the form $\tau \bowtie k$, where $k \in \mathbb{D}$ or is an instance-bound constant. Direct comparisons of the form $\tau_1 \bowtie \tau_2$ are excluded from scenario theories to prevent otherwise unconstrained parameters from supporting one another [7]; they may nevertheless occur in schema-induced expressions or fixed modelling axiom schemas, as exemplified later.

Set $\mathcal{C}_{\mathcal{R}}$ consists of spatial atoms

$$r_{\mathfrak{g}_1 \ldots \mathfrak{g}_n}(\bigcirc^{\ell_1} o_1, \ldots, \bigcirc^{\ell_n} o_n),$$

where $r_{\mathfrak{g}_1 \ldots \mathfrak{g}_n} \in \mathcal{R}$, $o_i \in \mathcal{O}_{\mathfrak{g}_i}$, and $\ell_i \in \mathbb{Z}$. Temporal shifts are interpreted according to the conventions introduced above, with $\bigcirc^0$ omitted.

A formula φ over the signature $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ is defined as

$$\varphi \ ::= c \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \bigcirc \varphi \mid \varphi \mathbb{U} \varphi \mid \varphi \mathbb{R} \varphi \mid \bullet \varphi \mid \varphi \mathbf{S} \varphi \mid \varphi \mathbf{T} \varphi, \qquad c \in \mathcal{C}.$$

Connectives $\bot$, $\wedge$, $\vee$ and $\to$ are *Boolean*, while the remaining connectives are *temporal* modalities. Hollow operators refer to future modalities and solid ones to past modalities. Accordingly, $\bigcirc$, $\mathbb{U}$, $\mathbb{R}$ represent the future modalities *next*, *until*, *release*, respectively, while $\bullet$, $\mathbf{S}$, $\mathbf{T}$ reflect their past counterparts: *previous*, *since*, *trigger*.

Boolean connectives are defined as $\neg\varphi \stackrel{def}{=} \varphi \to \bot$, $\top \stackrel{def}{=} \neg\bot$, $\varphi \leftrightarrow \psi \stackrel{def}{=} (\varphi \to \psi) \wedge (\psi \to \varphi)$, and derived temporal operators are defined as:

$$\begin{array}{llll} \mathbf{I} \stackrel{def}{=} \neg \bullet \top & \widehat{\bullet}\varphi \stackrel{def}{=} \bullet \varphi \vee \mathbf{I} & \blacksquare\varphi \stackrel{def}{=} \bot \mathbf{T} \varphi & \blacklozenge\varphi \stackrel{def}{=} \top \mathbf{S} \varphi \\ \mathbb{F} \stackrel{def}{=} \neg \bigcirc \top & \widehat{\bigcirc}\varphi \stackrel{def}{=} \bigcirc \varphi \vee \mathbb{F} & \square\varphi \stackrel{def}{=} \bot \mathbb{R} \varphi & \lozenge\varphi \stackrel{def}{=} \top \mathbb{U} \varphi \end{array}$$

Specifically, $\mathbf{I}$ and $\mathbb{F}$ refer to the "initial" and "final" state, respectively; $\bullet\,\varphi$ ($\bigcirc\,\varphi$) refers to φ at the previous (next) state in case it exists; $\blacksquare\,\varphi$ ($\square\,\varphi$) is read as "φ has always been true" ("φ will always be true"), and $\blacklozenge\,\varphi$ ($\lozenge\,\varphi$) is read as "φ has been true" ("φ will eventually be true").

A theory over a spatial domain $\mathcal{S}$ is a set of formulas over the signature induced by $\mathcal{S}$; we write it as $\Gamma^{\mathcal{S}}$ when the underlying spatial domain is relevant. As an example, let $\Gamma^{\mathcal{S}}$ be the growing-circle theory formalising specifications (1)–(4):

$$radius_c = 1.5 \ \wedge \ height_d = 5 \tag{1}$$

$$right_{\mathsf{pc}}(a, c) \ \wedge \ left_{\mathsf{pr}}(a, d) \ \wedge \ aligned Y_{\mathsf{pc}}(a, c) \tag{2}$$

$$aligned Y_{\mathsf{pp}}(b, a) \ \wedge \ left_{\mathsf{pp}}(\bigcirc b, a) \tag{3}$$

$$\square\big(p \ \to \ \big(grows_{\mathsf{c}}(c) \ \mathbb{U} \ \neg\neg extConnected_{\mathsf{cr}}(c, d)\big)\big) \tag{4}$$

Formula (3) uses an argument-level shift: $\mathit{left}_{\mathsf{pp}}(\bigcirc b, a)$ evaluates the parameters of b at the next state against the current parameters of a. By contrast, $\bigcirc\, \mathit{left}_{\mathsf{pp}}(b, a)$ would place the entire relation at the next state and would therefore involve next-state parameters for both objects. Finally, the endpoint in Formula (4) is double-negated. Since $\neg\varphi \stackrel{def}{=} \varphi \to \bot$, the expression $\neg\neg\alpha$ is equivalently $\neg\alpha \to \bot$: it requires α to hold without itself establishing the parameter values that make α true, following the same use of double negation for indirect satisfaction as in [8].

3.2. Semantics

Valuations and traces. A *valuation* over $\mathcal{X}$ and $\mathcal{D}$ is a function

$$v : \mathcal{X} \to \mathcal{D}_{\mathfrak{u}},$$

where $\mathcal{D}_{\mathfrak{u}} = \mathcal{D} \cup \{\mathfrak{u}\}$ and $\mathfrak{u}$ stands for *undefined*. Valuations respect the intended domains: propositional variables range over $\{\mathfrak{t}, \mathfrak{u}\}$, and numeric parameters over $\mathbb{D} \cup \{\mathfrak{u}\}$.

For instance, a valuation over the objects of the growing-circle example (not assumed at this stage to satisfy any formula) may be given by

$$v = \{\, x_c \mapsto 4,\ y_c \mapsto \mathfrak{u},\ \mathit{radius}_c \mapsto 1,\ p \mapsto \mathfrak{t},\ \ldots \},$$

where the ellipsis stands for assignments to the remaining parameter variables of objects a, b and d. This valuation fixes the x-coordinate and the radius of circle c, leaves its y-coordinate undefined, and records the truth of the propositional atom p. We write $\mathcal{V}$ for the set of all valuations.

A valuation v can also be viewed as a set of pairs $\{x \mapsto v(x) \mid x \in \mathcal{X}, v(x) \in \mathcal{D}\}$, which drops any pairs $x \mapsto \mathfrak{u}$ for $x \in \mathcal{X}$. For instance, the valuation above can equivalently be written as $v = \{\, (x_c, 4),\ (\mathit{radius}_c, 1),\ (p, \mathfrak{t}),\ \ldots \}$, where pairs involving variables mapped to $\mathfrak{u}$ are omitted. This allows using standard set inclusion for comparison where $v \subseteq v'$ stands for $\{x \mapsto v(x) \mid x \in \mathcal{X},\, v(x) \in \mathcal{D}\} \subseteq \{x \mapsto v'(x) \mid x \in \mathcal{X},\, v'(x) \in \mathcal{D}\}$. This is equivalent to: $v(x) \in \mathcal{D}$ implies $v'(x) = v(x)$ for all $x \in \mathcal{X}$.

A *trace* $\boldsymbol{v}$ of length λ, where λ is supplied by the instance $\mathcal{I}$, is a finite sequence $\boldsymbol{v} = (v_i)_{i \in [0..\lambda)}$ of valuations. We restrict attention to finite traces of fixed length λ, which suffice for the scenario-based and diagrammatic reasoning considered here. For example, a trace of length $\lambda = 2$ may be given by

$$\boldsymbol{v} = \{\, x_a \mapsto 1,\ y_a \mapsto \mathfrak{u},\ \ldots \} \cdot \{\, x_a \mapsto 3,\ y_a \mapsto 2,\ \ldots \},$$

where only the assignments of point a are shown. We denote by $\boldsymbol{\mathcal{V}}_{\mathcal{X},\mathcal{D}}$ the set of all such traces over $\mathcal{X}$ and $\mathcal{D}$, and remove the subindices when clear from context. The relation $\subseteq$ is extended pointwise to traces: given two traces $\boldsymbol{v} = (v_i)_{i \in [0..\lambda)}$ and $\boldsymbol{v}' = (v'_i)_{i \in [0..\lambda)}$ of length λ, we write $\boldsymbol{v} \subseteq \boldsymbol{v}'$ if $v_i \subseteq v'_i$ for all $i \in [0..\lambda)$.

Denotation of atoms. Atomic satisfaction is defined by a time-indexed denotation[1]

$$[\![\cdot]\!]_i : \mathcal{C} \to 2^{\boldsymbol{\mathcal{V}}}, \qquad i \in [0..\lambda).$$

For every regular atom

$$\alpha \in \mathcal{C}_{\mathcal{P}} \cup \mathcal{C}_{\mathcal{A}},$$

the denotation is time-local and given by

$$[\![\alpha]\!]_i \stackrel{def}{=} \{\boldsymbol{v} \in \boldsymbol{\mathcal{V}} \mid v_i(\alpha) = \mathfrak{t}\}.$$

Thus, ordinary propositional atoms and action atoms have the same regular-atom denotation, although they remain syntactically distinguished. An action atom records the occurrence of an action at the

[1] Standard THTc [6] uses solution relations over tuples of values. The equivalent time-indexed presentation adopted here is closer to [7] and makes temporal shifts explicit.

current state; its connection with the corresponding effect is introduced later through the formulas induced by effect.

Derived-quantities are expanded into their defining expressions over parameters, with any temporal shift applied uniformly to all parameters in the expansion. Each valuation v_i is then extended to shifted parameters by

$$v_i(\bigcirc^{\ell} x) \stackrel{def}{=} \begin{cases} v_{i+\ell}(x), & i+\ell \in [0..\lambda), \\ \mathfrak{u}, & \text{otherwise.} \end{cases}$$

This evaluation extends compositionally to any arithmetic expressions, with undefined values propagated through arithmetic operations.

For a numerical atom $\tau_1 \bowtie \tau_2 \in \mathcal{C}_{\mathcal{N}}$, satisfaction follows its usual mathematical meaning and requires all parameter variables needed to evaluate the expressions to be defined at their respective time steps:

$$[\![\tau_1 \bowtie \tau_2]\!]_i \stackrel{def}{=} \{\boldsymbol{v} \in \boldsymbol{\mathcal{V}} \mid v_i(\tau_1), v_i(\tau_2) \in \mathbb{D} \text{ and } v_i(\tau_1) \bowtie v_i(\tau_2)\}.$$

For example, in the growing-circle scenario,

$$[\![\mathit{radius}_c = 1.5]\!]_i \stackrel{def}{=} \{\boldsymbol{v} \in \boldsymbol{\mathcal{V}} \mid v_i(\mathit{radius}_c) = 1.5\}.$$

For a derived-quantity, its defining expression is evaluated instead; for instance,

$$[\![\mathit{rightSide}_d = 3]\!]_i \stackrel{def}{=} \left\{ \boldsymbol{v} \in \boldsymbol{\mathcal{V}} \;\middle|\; \begin{array}{l} v_i(\mathit{leftSide}_d), v_i(\mathit{width}_d) \in \mathbb{R}, \\ v_i(\mathit{leftSide}_d) + v_i(\mathit{width}_d) = 3 \end{array} \right\}.$$

For spatial atoms from $\mathcal{C}_{\mathcal{R}}$, let

$$\alpha = r_{\mathfrak{g}_1 \ldots \mathfrak{g}_n}(\bigcirc^{\ell_1} o_1, \ldots, \bigcirc^{\ell_n} o_n) \in \mathcal{C}_{\mathcal{R}}.$$

Its numerical characterisation κ_α is obtained from $\mathrm{constr}(r_{\mathfrak{g}_1 \ldots \mathfrak{g}_n})[o_1, \ldots, o_n]$ by expanding derived-quantities and shifting every parameter associated with the j-th argument by ℓ_j. Similarly to numerical atoms, all parameter values occurring in the instantiated constraint must be defined.[2]

For instance, if

$$\mathrm{constr}(\mathit{left}_{\mathsf{pr}}) \stackrel{def}{=} x_P < \mathit{leftSide}_R,$$

then instantiating this constraint schema with a and d yields

$$\mathrm{constr}(\mathit{left}_{\mathsf{pr}})[a, d] \stackrel{def}{=} x_a < \mathit{leftSide}_d,$$

and this instantiated constraint can then be used to define the denotation

$$[\![\mathit{left}_{\mathsf{pr}}(a, d)]\!]_i \stackrel{def}{=} \left\{ \boldsymbol{v} \in \boldsymbol{\mathcal{V}} \;\middle|\; v_i(x_a), v_i(\mathit{leftSide}_d) \in \mathbb{R} \text{ and } v_i(x_a) < v_i(\mathit{leftSide}_d) \right\}.$$

Negative spatial atoms are interpreted analogously through their corresponding constraint schemas.

For shifted arguments, parameters are evaluated at the times determined by their shifts. Thus, if

$$\mathrm{constr}(\mathit{left}_{\mathsf{pp}}) \stackrel{def}{=} x_{P_1} < x_{P_2},$$

then

$$[\![\mathit{left}_{\mathsf{pp}}(\bigcirc b, a)]\!]_i \stackrel{def}{=} \{\boldsymbol{v} \in \boldsymbol{\mathcal{V}} \mid v_i(\bigcirc x_b), v_i(x_a) \in \mathbb{R} \text{ and } v_i(\bigcirc x_b) < v_i(x_a)\},$$

so the x-coordinate of the first argument is read from the next time step.

[2] Note that, unlike the spatial atoms, the numerical atoms defined previously require no separate instantiation, since they are already object-language expressions over concrete parameter variables.

Satisfaction. We now define when a formula holds at a given time step in an interpretation. An *interpretation* of length λ is a pair $\langle \boldsymbol{h}, \boldsymbol{t} \rangle$ of traces of length λ such that $\boldsymbol{h} \subseteq \boldsymbol{t}$. It is *total* when $\boldsymbol{h} = \boldsymbol{t}$. Since we only consider strict atoms, that is, atoms whose variables must all be defined in order for the atom to hold, atomic satisfaction depends solely on the $\boldsymbol{h}$ component of the interpretation [6].

Given an interpretation $\langle \boldsymbol{h}, \boldsymbol{t} \rangle$ of length λ, a time step $i \in [0..\lambda)$, and a constraint atom $c \in \mathcal{C}$,

$$\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models c \quad \text{iff} \quad \boldsymbol{h} \in [\![c]\!]_i.$$

The satisfaction clauses for the Boolean and temporal operators are given below and follow the definitions in [6].

1. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \varphi \wedge \psi$ *if* $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \varphi$ *and* $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \psi$.
2. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \varphi \vee \psi$ *if* $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \varphi$ *or* $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \psi$.
3. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \varphi \rightarrow \psi$ *if* $\big(\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \not\models \varphi$ or $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \psi\big)$ and $\big(\langle \boldsymbol{t}, \boldsymbol{t} \rangle, i \not\models \varphi$ or $\langle \boldsymbol{t}, \boldsymbol{t} \rangle, i \models \psi\big)$.
4. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \bigcirc \varphi$ *if* $i < \lambda - 1$ *and* $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i + 1 \models \varphi$.
5. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \varphi \,\mathbb{U}\, \psi$ *if* there exists $k \in [i..\lambda)$ s.t. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, k \models \psi$ and $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, j \models \varphi$ for all $i \leq j < k$.
6. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \varphi \,\mathbb{R}\, \psi$ *if* for all $k \in [i..\lambda)$, either $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, k \models \psi$ or $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, j \models \varphi$ for some $i \leq j < k$.
7. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \bullet \varphi$ *if* $i > 0$ *and* $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i - 1 \models \varphi$.
8. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \varphi \,\mathbf{S}\, \psi$ *if* there exists $k \in [0..i]$ s.t. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, k \models \psi$ and $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, j \models \varphi$ for all $k < j \leq i$.
9. $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \varphi \,\mathbf{T}\, \psi$ *if* for all $k \in [0..i]$, either $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, k \models \psi$ or $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, j \models \varphi$ for some $k < j \leq i$.

For a theory Γ, we write

$$\langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \Gamma \quad \text{iff} \quad \langle \boldsymbol{h}, \boldsymbol{t} \rangle, i \models \varphi \text{ for every } \varphi \in \Gamma.$$

Definition 1 (Equilibrium model and stable model)**.** *A total interpretation $\langle \boldsymbol{t}, \boldsymbol{t} \rangle$ of length λ is an* equilibrium model *of Γ if $\langle \boldsymbol{t}, \boldsymbol{t} \rangle, 0 \models \Gamma$ and there is no $\boldsymbol{h} \subset \boldsymbol{t}$ such that $\langle \boldsymbol{h}, \boldsymbol{t} \rangle, 0 \models \Gamma$. We call $\boldsymbol{t}$ a* stable model *of Γ and denote the set of stable models of length λ by $\mathrm{SM}_\lambda(\Gamma)$.*

3.3. Spatio-Temporal Background Axioms

Besides interpreting spatial atoms through constr, a spatial domain induces formulas that apply uniformly to every scenario theory over that domain. These comprise structural admissibility conditions obtained from degen and lim, as well as the effects associated with action occurrences through effect. Their schemas are instantiated using the objects and bounds supplied by the instance.

We define

$$\mathrm{Ax}_{\mathcal{S}}^{\mathrm{degen}} \stackrel{def}{=} \bigcup_{\mathfrak{g} \in \mathcal{G}} \bigcup_{o \in \mathcal{O}_{\mathfrak{g}}} \big\{ \Box\big(\mathrm{degen}(\mathfrak{g})[o] \rightarrow \bot\big) \big\},$$

$$\mathrm{Ax}_{\mathcal{S}}^{\mathrm{lim}} \stackrel{def}{=} \bigcup_{\mathfrak{g} \in \mathcal{G}} \bigcup_{o \in \mathcal{O}_{\mathfrak{g}}} \big\{ \Box\big(\mathrm{lim}(\mathfrak{g})[o] \rightarrow \bot\big) \big\},$$

$$\mathrm{Ax}_{\mathcal{S}}^{\mathrm{effect}} \stackrel{def}{=} \bigcup_{a_{\mathfrak{g}_1 \ldots \mathfrak{g}_n} \in \mathcal{A}} \bigcup_{\substack{(o_1, \ldots, o_n) \in \\ \mathcal{O}_{\mathfrak{g}_1} \times \cdots \times \mathcal{O}_{\mathfrak{g}_n}}} \Big\{ \Box\Big(a_{\mathfrak{g}_1 \ldots \mathfrak{g}_n}(o_1, \ldots, o_n) \rightarrow \mathrm{effect}(a_{\mathfrak{g}_1 \ldots \mathfrak{g}_n})[o_1, \ldots, o_n] \Big) \Big\}.$$

Collecting these components, the background axiom set of $\mathcal{S}$ is

$$\mathrm{Ax}_{\mathcal{S}} \stackrel{def}{=} \mathrm{Ax}_{\mathcal{S}}^{\mathrm{degen}} \cup \mathrm{Ax}_{\mathcal{S}}^{\mathrm{lim}} \cup \mathrm{Ax}_{\mathcal{S}}^{\mathrm{effect}}.$$

For example, the circle degeneracy schema yields the following member of $\mathrm{Ax}_{\mathcal{S}}^{\mathrm{degen}}$:

$$\Box(\mathit{radius}_c \leq 0 \rightarrow \bot).$$

Likewise, for $I_x = [x_{\min}, x_{\max}]$ and $I_y = [y_{\min}, y_{\max}]$, instantiating the point limit schema with object a yields the following member of $\mathrm{Ax}_{\mathcal{S}}^{\mathrm{lim}}$:

$$\Box\big((x_a < x_{\min} \vee x_a > x_{\max} \vee y_a < y_{\min} \vee y_a > y_{\max}) \to \bot\big).$$

Because their numerical atoms are strict, the formulas in $\mathrm{Ax}_{\mathcal{S}}^{\mathrm{degen}}$ and $\mathrm{Ax}_{\mathcal{S}}^{\mathrm{lim}}$ exclude defined inadmissible configurations but do not themselves assign parameter values.

The effect schema for the growth action yields the following member of $\mathrm{Ax}_{\mathcal{S}}^{\mathrm{effect}}$:

$$\Box\Big(\mathit{grows}_{\mathrm{c}}(c) \to \bigcirc\, \mathit{radius}_c > \mathit{radius}_c\Big).$$

Consequently, the formulas in $\mathrm{Ax}_{\mathcal{S}}^{\mathrm{effect}}$ enforce the corresponding transition whenever an action occurrence is established.

The complete theory associated with a scenario theory is

$$\Gamma_*^{\mathcal{S}} \stackrel{def}{=} \Gamma^{\mathcal{S}} \cup \mathrm{Ax}_{\mathcal{S}}, \qquad \mathrm{SM}^{\mathcal{S}}(\Gamma^{\mathcal{S}}) \stackrel{def}{=} \mathrm{SM}_{\lambda}(\Gamma_*^{\mathcal{S}}),$$

where the finite trace length λ is supplied by the instance $\mathcal{I}$.

4. Modelling Spatio-Temporal Scenarios

We now turn from the induced spatio-temporal background axioms to optional modelling schemas for concrete spatio-temporal scenarios. Throughout this section, a scenario theory $\Gamma^{\mathcal{S}}$ is interpreted through $\Gamma_*^{\mathcal{S}}$. Unlike the background axioms, the following schemas are included only when required by the intended scenario.

4.1. Modelling Axiom Schemas

4.1.1. Definability Axioms

Parameter variables remain undefined unless their values are supported by spatial relations, assignments, action effects, or other formulas. Since equality is strict, definability can be expressed, for $x \in \mathcal{X}_{\mathrm{param}}$ and $o \in \mathcal{O}_{\mathfrak{g}}$ with $\mathfrak{g} \in \mathcal{G}$, by the abbreviations

$$\mathit{df}(x) \stackrel{def}{=} x = x, \qquad \mathit{df}(o) \stackrel{def}{=} \bigwedge_{x \in \mathcal{X}_{\mathrm{param}}|_o} \mathit{df}(x).$$

When all objects must be fully specified initially, we use

$$\mathit{df}(o) \qquad \text{for every } \mathfrak{g} \in \mathcal{G} \text{ and } o \in \mathcal{O}_{\mathfrak{g}}. \tag{df}$$

Since theories are evaluated at time 0, this is an initialisation condition; definedness at later states must be supported by inertia, action effects, or other formulas.

4.1.2. Inertia Axioms

Parameter values persist by default through the schema

$$\Box\Big((\neg\mathbb{F} \wedge \mathit{df}(x) \wedge \neg(\bigcirc x \neq x)) \to \bigcirc x = x\Big) \quad \begin{array}{l}\text{for every } \mathfrak{g} \in \mathcal{G},\ o \in \mathcal{O}_{\mathfrak{g}},\\ \text{and } x \in \mathcal{X}_{\mathrm{param}}|_o.\end{array} \tag{inertia}$$

The guard $\neg\mathbb{F}$ restricts the axiom to states with a successor. If x is undefined, inertia imposes no requirement; otherwise its current value is propagated unless a different next value can be established. The shifted comparison is permitted here as part of a fixed axiom schema. Requiring $\Box\mathit{df}(o)$, rather than the initial condition above, would provide fresh support at every state and may defeat default persistence.

4.2. Scenarios and Diagrams

From each stable model, we extract a projected representation consisting of two components: a qualitative component, called a *scenario*, and a numerical component, called a *diagrammatic specification*. A scenario records the propositional and spatial atoms satisfied at each state, forming a *scene*, together with the action occurrences labelling the transitions between consecutive scenes. The diagrammatic specification records the parameter values that realise the spatial atoms geometrically.

In usual HTc-based semantics with mathematical theories, theory atoms such as $x > y$ merely restrict admissible valuations and are not reported in the resulting answers. In the present setting, however, spatial atoms constitute qualitative information of interest. We therefore retain spatial atoms as state labels and action atoms as transition labels.

Definition 2 (Answer set induced by a stable model)**.** *Let* $\boldsymbol{t} = (t_i)_{i\in[0..\lambda)} \in \mathrm{SM}^{\mathcal{S}}(\Gamma^{\mathcal{S}})$. *The* answer set *induced by* $\boldsymbol{t}$ *is*

$$\big(\mathrm{scen}(\boldsymbol{t}), \mathrm{diag}(\boldsymbol{t})\big) \stackrel{def}{=} \big((\boldsymbol{\sigma}, \boldsymbol{\alpha}), \boldsymbol{\delta}\big),$$

where the sequence of scenes is

$$\boldsymbol{\sigma} = (\sigma_i)_{i\in[0..\lambda)}, \qquad \sigma_i \stackrel{def}{=} \{c \in \mathcal{C}_{\mathcal{P}} \cup \mathcal{C}_{\mathcal{R}} \mid \boldsymbol{t} \in [\![c]\!]_i\},$$

the sequence of transition labels is

$$\boldsymbol{\alpha} = (\alpha_i)_{i\in[0..\lambda-1)}, \qquad \alpha_i \stackrel{def}{=} \{c \in \mathcal{C}_{\mathcal{A}} \mid \boldsymbol{t} \in [\![c]\!]_i\},$$

and the sequence of diagrammatic state specifications is

$$\boldsymbol{\delta} = (\delta_i)_{i\in[0..\lambda)}, \qquad \delta_i \stackrel{def}{=} \{x \mapsto t_i(x) \mid x \in \mathcal{X}_{\mathrm{param}},\ t_i(x) \in \mathbb{D}\},$$

for each $i \in [0..\lambda)$. *We call* $(\boldsymbol{\sigma}, \boldsymbol{\alpha})$ *the induced* scenario *and* $\boldsymbol{\delta}$ *the induced* diagrammatic specification.

A scenario can thus be displayed as the alternating sequence

$$\sigma_0 \xrightarrow{\alpha_0} \sigma_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{\lambda-2}} \sigma_{\lambda-1}.$$

Although an action atom in α_i is semantically evaluated at state i, it is displayed between states i and $i+1$ because its effect constrains that transition. Numerical atoms are omitted, while the diagrammatic specification $\boldsymbol{\delta}$ records the parameter values realising the spatial component of the scenario.

A fixed scenario may correspond to several diagrammatic specifications, induced by different stable models $\boldsymbol{t}$ satisfying

$$\mathrm{scen}(\boldsymbol{t}) = (\boldsymbol{\sigma}, \boldsymbol{\alpha}).$$

Such specifications may be partial when some parameters remain undefined; when all parameters are defined at every state, they can be rendered as concrete diagrams. For representation purposes, one specification is usually sufficient.

Example. We now consider the complete growing-circle theory $\Gamma^{\mathcal{S}}_{*}$ extended with the modelling schemas for inertia and initial definability, written $\Gamma^{\mathcal{S}}_{\{\mathrm{inertia,df}\}}$, together with the assumption that p holds. A stable model $\boldsymbol{t} = (t_0, t_1)$ of this theory may, for instance, contain

$$\begin{aligned} t_0 =& \{\, \mathit{radius}_c \mapsto 1.5,\ \mathit{leftSide}_d \mapsto 5.7,\ x_b \mapsto 8.4,\ y_b \mapsto 4,\ p \mapsto \mathfrak{t},\ \mathit{grows}_{\mathrm{c}}(c) \mapsto \mathfrak{t},\ \ldots\}, \\ t_1 =& \{\, \mathit{radius}_c \mapsto 2.7,\ \mathit{leftSide}_d \mapsto 5.7,\ x_b \mapsto 2.4,\ y_b \mapsto 4,\ \ldots\}. \end{aligned}$$

The dots indicate further parameter assignments required by the spatial relations, action effects, and modelling axioms.

Let

$$(\boldsymbol{\sigma}, \boldsymbol{\alpha}) \stackrel{def}{=} \mathrm{scen}(\boldsymbol{t}) \qquad \text{and} \qquad \boldsymbol{\delta} \stackrel{def}{=} \mathrm{diag}(\boldsymbol{t}).$$

In practice, it is often more informative to display only atoms that occur in the theory, possibly together with selected consequences of interest, rather than the full projection over all atoms in $\mathcal{C}_{\mathcal{P}} \cup \mathcal{C}_{\mathcal{R}}$. We follow this convention below.

The sequence of scenes $\boldsymbol{\sigma} = (\sigma_0, \sigma_1)$ is displayed as follows:

$$\sigma_0 = \{\, p,\ \mathit{right}_{\mathsf{pc}}(a,c),\ \mathit{left}_{\mathsf{pr}}(a,d),\ \mathit{left}_{\mathsf{pp}}(\circ b, a),\ \ldots\},$$
$$\sigma_1 = \{\, \mathit{inside}_{\mathsf{pc}}(a,c),\ \mathit{left}_{\mathsf{pp}}(b,a),\ \mathit{extConnected}_{\mathsf{cr}}(c,d),\ \ldots\}.$$

Here, the dots stand only for further atoms from the selected display vocabulary that hold at the corresponding state. The transition-label sequence is

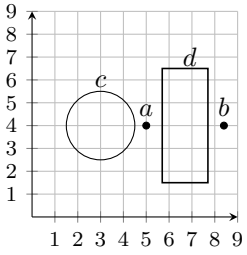
$$\boldsymbol{\alpha} = (\alpha_0), \qquad \alpha_0 = \{\, \mathit{grows}_{\mathsf{c}}(c)\,\}.$$

Thus, the scenario may be displayed as

$$\sigma_0 \xrightarrow{\alpha_0} \sigma_1.$$

The associated diagrammatic specification $\boldsymbol{\delta} = (\delta_0, \delta_1)$ is given together with its rendering in Figure 1.

$$\delta_0 = \left\{ \begin{array}{lll} x_a \mapsto 5 & y_a \mapsto 4 & \\ x_b \mapsto 8.4 & y_b \mapsto 4 & \\ x_c \mapsto 3 & y_c \mapsto 4 & \mathit{radius}_c \mapsto 1.5 \\ \mathit{leftSide}_d \mapsto 5.7 & \mathit{bottom}_d \mapsto 1.5 & \\ \mathit{width}_d \mapsto 2 & \mathit{height}_d \mapsto 5 & \end{array} \right\}$$



$$\downarrow$$

$$\delta_1 = \left\{ \begin{array}{lll} x_a \mapsto 5 & y_a \mapsto 4 & \\ x_b \mapsto 2.4 & y_b \mapsto 4 & \\ x_c \mapsto 3 & y_c \mapsto 4 & \mathit{radius}_c \mapsto 2.7 \\ \mathit{leftSide}_d \mapsto 5.7 & \mathit{bottom}_d \mapsto 1.5 & \\ \mathit{width}_d \mapsto 2 & \mathit{height}_d \mapsto 5 & \end{array} \right\}$$

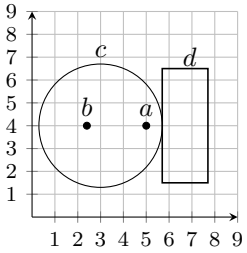


Figure 1: Diagrammatic specification $\boldsymbol{\delta}$ and its rendering at time steps 0 and 1 for the growing-circle example.

The initial definability schema supplies values for otherwise unconstrained parameters, such as the width of rectangle d. At time 0, $\mathit{left}_{\mathsf{pp}}(\circ b, a)$ constrains x_b at time 1 relative to the current value of x_a, while inertia supports the persistence of the remaining parameters. The occurrence $\mathit{grows}_{\mathsf{c}}(c)$ imposes

$$\circ\, \mathit{radius}_c > \mathit{radius}_c.$$

The double-negated endpoint does not itself generate a geometric configuration; it holds once the resulting values realise $\mathit{extConnected}_{\mathsf{cr}}(c,d)$. In the displayed stable model, circle c therefore grows from radius 1.5 to 2.7, while point b lies to the left of point a at the next state.

Other stable models may induce different initial scenes. For example, point b may already lie inside circle c at time 0, since the initial theory constrains only its vertical alignment with point a. The induced scenario would then differ from $(\boldsymbol{\sigma}, \boldsymbol{\alpha})$ by containing $\mathit{inside}_{\mathsf{pc}}(b,c)$ and $\mathit{left}_{\mathsf{pr}}(b,d)$ at time 0. This also makes explicit the role of the chosen spatial vocabulary: scenarios are not absolute descriptions of a configuration, but projections of stable models relative to the spatial atoms in $\mathcal{C}_{\mathcal{R}}$.

The example also exhibits spatial ramification. At time 1, the increased radius makes a lie inside c, although $\mathit{inside}_{\mathsf{pc}}(a,c)$ is not stated as an action effect, and the required external connection appears once its denotation becomes true. Likewise, relations such as $\mathit{alignedY}_{\mathsf{pc}}(a,c)$ may persist because their underlying parameters persist. Thus, inertia operates on parameter values, while spatial relations are recovered from those values, avoiding separate ramification and persistence rules for each relation.

5. Discussion and Related Work

The framework follows the declarative spatial reasoning line initiated by CLP(QS) [1] and continued in ASPMT(QS) [3]: spatial relations are interpreted by constraints over geometric parameters rather than treated as uninterpreted symbols. This preserves the main advantage of DSR, namely, that qualitative relations interact through their shared numeric realisations. As proved in [9, 10, 11], many relations from standard qualitative calculi admit such encodings by quantifier-free polynomial constraints, including relations from Allen's Interval Algebra [12], Rectangle Algebra [13], the Left–Right Algebra [14], RCC fragments over suitable geometric domains [15, 16], and the Cardinal Direction Calculus [17]. In this sense, the framework does not commit to a single calculus or spatial aspect. Typed spatial atoms provide a uniform vocabulary in which topological, directional, metric, and application-specific relations across heterogeneous geometric sorts can be introduced as needed.

Compared with the semantics underlying ASPMT(QS), the present framework is located in a different logical setting. ASPMT(QS) is based on ASPMT and uses completion-based techniques for tight instances, translating them into SMT instances in order to compute stable models [18]. The present framework is instead formulated directly in THTc. This provides a single logical semantics in which spatial constraints, temporal operators, partial valuations, and stable-model minimisation interact explicitly, rather than being mediated by a computational translation.

Partial valuations are another point of departure from standard encodings. They make it possible to leave parameters undefined unless they are constrained by the theory, and to formulate inertia uniformly at the level of parameter variables. This avoids encoding persistence through families of object-specific time-indexed fluents and associated state constraints, as is common in ASP planning encodings [19]. As a consequence, definability becomes a modelling choice: depending on the intended task, one may require complete diagrammatic witnesses or allow partial specifications. This is particularly useful for scenario-based reasoning, where a qualitative temporal evolution may be meaningful even when only some of its numeric parameters need to be fixed.

The use of geometric sorts also relates the framework to sorted extensions of ASP. Languages such as SPARC [20] provide explicit constructs for specifying objects, relations, and their sorts, thereby improving program structure and enabling type-related checks. Our use of sorts has a different role: sorts determine the geometric parameters associated with an object and the admissible types of spatial relations. This is important for heterogeneous spatial reasoning, where points, circles, rectangles, polygons, and other geometric objects may occur in the same theory while being governed by different parameterisations and well-formedness conditions.

The present paper focuses on a small set of geometric sorts, but the same semantic pattern extends naturally to higher-dimensional and richer spatial domains. General polygons, for example, can be represented by vertex parameters. This also suggests a more systematic order-sorted treatment of geometric objects, where specific shapes such as rectangles or triangles are subsumed under more general polygonal sorts [21]. On the logical side, another natural extension is to add aggregate constructs to THTc, following existing work on aggregates in equilibrium logic and ASP with constraints [22, 23]. Aggregates would be useful for expressing spatial conditions involving collections of objects, such as counting contacts, bounding total occupied area, or defining support conditions.

Even at the semantic level, the framework can serve as a formal basis for cognitive modelling approaches concerned with schematic mental representations [24]. Executable specialisations can also support action and planning formalisms for spatial domains [19], diagrammatic reasoning [25], and spatial reasoning with embodied language models [26]. The modularity of the framework permits different implementation strategies depending on the mathematical language underlying the spatial domain. For linear integer constraints, the present approach can exploit meta-programming with *metasp* [27] and use *flingo* [28] as a backend, whose founded variables support partial numerical valuations. Real-valued or nonlinear domains may instead require SMT-based systems such as Z3 [29] or IDP-Z3 [30], integrated with the *clingo* Python API in a multi-shot setting for temporal reasoning [31, 32]. Finally, EZSMT [33], which directly combines answer-set semantics with integer and real constraints, may provide another suitable backend for a meta-programming implementation.

Declaration on Generative AI

Generative AI was used exclusively for grammatical and language editing of this manuscript. All content was reviewed and edited by the authors, who take full responsibility for it. All ideas and scientific content are the authors' own.

References

[1] M. Bhatt, J. H. Lee, C. Schultz, CLP(QS): A declarative spatial reasoning framework, in: Spatial Information Theory: 10th International Conference, COSIT 2011, Belfast, ME, USA, September 12-16, 2011. Proceedings. Lecture Notes in Computer Science, volume 6899, Springer, 2011, pp. 210–230.

[2] G. Ligozat, Qualitative spatial and temporal reasoning, John Wiley & Sons, 2013.

[3] P. A. Wałęga, C. Schultz, M. Bhatt, Non-monotonic spatial reasoning with answer set programming modulo theories, Theory and Practice of Logic Programming 17 (2017) 205–225. doi:10.1017/S1471068416000193.

[4] C. Schultz, M. Bhatt, J. Suchan, P. A. Wałęga, Answer set programming modulo 'space-time', in: International Joint Conference on Rules and Reasoning, Springer, 2018, pp. 318–326.

[5] B. Li, C. Schultz, Clingo2dsr-a clingo-based software system for declarative spatial reasoning, Spatial Cognition & Computation (2024) 1–51.

[6] P. Cabalar, M. Diéguez, F. Olivier, T. Schaub, I. Stephan, Towards constraint temporal answer set programming, Theory and Practice of Logic Programming (2025).

[7] P. Cabalar, R. Kaminski, M. Ostrowski, T. Schaub, An asp semantics for default reasoning with constraints., in: IJCAI, 2016, pp. 1015–1021.

[8] A. Becker, P. Cabalar, M. Diéguez, S. Hahn, J. Romero, T. Schaub, Implementing metric temporal answer set programming, 2026. URL: https://arxiv.org/abs/2601.20735. arXiv:2601.20735, cs.LO.

[9] G. Pesant, M. Boyer, Quad-clp (r): Adding the power of quadratic constraints, in: International Workshop on Principles and Practice of Constraint Programming, Springer, 1994, pp. 95–108.

[10] D. Bouhineau, Solving geometrical constraint systems using clp based on linear constraint solver, in: International Conference on Artificial Intelligence and Symbolic Mathematical Computing, Springer, 1996, pp. 274–288.

[11] G. Pesant, M. Boyer, Reasoning about solids using constraint logic programming, Journal of Automated Reasoning 22 (1999) 241–262.

[12] J. F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26 (1983) 832–843.

[13] H. W. Guesgen, Spatial Reasoning Based on Allen's Temporal Logic, Technical Report TR-89-049, International Computer Science Institute, Berkeley, 1989.

[14] A. Scivos, B. Nebel, The finest of its class: The natural point-based ternary calculus for qualitative spatial reasoning, in: International Conference on Spatial Cognition, Springer, 2004, pp. 283–303.

[15] D. A. Randell, Z. Cui, A. G. Cohn, A spatial logic based on regions and connection., KR 92 (1992) 165–176.

[16] I. Düntsch, H. Wang, S. McCloskey, Relations algebras in qualitative spatial reasoning, Fundamenta Informaticae 39 (1999) 229–248.

[17] A. U. Frank, Qualitative spatial reasoning with cardinal directions, in: 7. Österreichische Artificial-Intelligence-Tagung/Seventh Austrian Conference on Artificial Intelligence, Springer, 1991, pp. 157–167.

[18] M. Bartholomew, J. Lee, Functional stable model semantics and answer set programming modulo theories, in: Twenty-Third International Joint Conference on Artificial Intelligence, 2013.

[19] F. Ahmad, V. Patoglu, E. Erdem, Hybrid planning for challenging construction problems: An answer set programming approach, Artificial Intelligence 319 (2023) 103902.

[20] E. Balai, M. Gelfond, Y. Zhang, Towards answer set programming with sorts, in: International Conference on Logic Programming and Nonmonotonic Reasoning, Springer, 2013, pp. 135–147.
[21] K. Kaneiwa, Order-sorted logic programming with predicate hierarchy, Artificial Intelligence 158 (2004) 155–188.
[22] P. Cabalar, J. Fandinno, T. Schaub, P. Wanko, A uniform treatment of aggregates and constraints in hybrid asp, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, volume 17, 2020, pp. 193–202.
[23] P. Cabalar, J. Fandinno, T. Schaub, P. Wanko, An asp semantics for constraints involving conditional aggregates, in: ECAI 2020, IOS Press, 2020, pp. 664–671.
[24] F. Olivier, Z. Bouraoui, Grounding agent reasoning in image schemas: A neurosymbolic approach to embodied cognition, in: Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), IFAAMAS, 2025, pp. 2875–2879. URL: https://www.ifaamas.org/Proceedings/aamas2025/pdfs/p2875.pdf.
[25] D. Wang, M. Jamnik, P. Lio, Abstract diagrammatic reasoning with multiplex graph networks, arXiv preprint arXiv:2006.11197 (2020).
[26] F. Olivier, Z. Bouraoui, Towards a neurosymbolic reasoning system grounded in schematic representations, in: Proceedings of the 19th International Conference on Neurosymbolic Learning and Reasoning (NeSy 2025), 2025.
[27] S. Hahn, A. Nemes, J. Romero, T. Schaub, Meta-programming for linear-time temporal answer set programming, arXiv preprint arXiv:2605.29965 (2026). URL: https://arxiv.org/abs/2605.29965.
[28] J. Fandinno, P. Cabalar, P. Wanko, T. Schaub, Flingo – instilling asp expressiveness into linear integer constraints, 2026. URL: https://arxiv.org/abs/2602.09620. arXiv:2602.09620.
[29] L. de Moura, N. Bjørner, Z3: An efficient smt solver, in: Proceedings of the Theory and Practice of Software (TACAS), Springer, 2008, pp. 337–340.
[30] P. Carbonnelle, S. Vandevelde, J. Vennekens, M. Denecker, Idp-z3: a reasoning engine for fo(.), 2022. URL: https://arxiv.org/abs/2202.00343. arXiv:2202.00343.
[31] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot asp solving with clingo, Theory and Practice of Logic Programming 19 (2019) 27–82.
[32] R. Kaminski, J. Romero, T. Schaub, P. Wanko, How to build your own asp-based system?!, Theory and Practice of Logic Programming 23 (2023) 299–361.
[33] K. Dhakal, Y. Lierler, Ezsmt version 3, matured, arXiv preprint arXiv:2607.13344 (2026). Under consideration in Theory and Practice of Logic Programming (TPLP).