H-Complexity Metric for Qualitative Spatial and Temporal Reasoning Applications

Carl Schultz, Robert Amor
Department of Computer Science, The University of Auckland
Private Bag 92019, Auckland, New Zealand
csch050@aucklanduni.ac.nz, trebor@cs.auckland.ac.nz

Hans Guesgen
School of Engineering and Advanced Technology, Massey University
Private Bag 11222, Palmerston North, New Zealand
h.w.guesgen@massey.ac.nz

Abstract

Commonsense reasoning, in particular qualitative spatial and temporal reasoning (QSTR), provides flexible and intuitive methods for reasoning about vague and uncertain information including temporal duration and ordering, and spatial orientation, topology and distance. Despite significant theoretical advances in QSTR, there is a distinct absence of applications that employ these methods. The central problem is a lack of application-level standards and metrics that developers can use to measure the effectiveness of their QSTR applications. To address this we present a fundamental metric called *H*-complexity that quantifies the expressiveness of QSTR systems according to the number of distinct scenario classes that can be encoded. In this paper we show that *H*-complexity can be employed in a range of powerful and practical ways that support QSTR application development. To illustrate this, we present two examples: calculating test coverage for validation, and quantifying the reduction in expressiveness due to constraints. We thereby demonstrate that *H*-complexity is a useful tool for determining whether a QSTR system meets the needs of a specific application.

Introduction

Commonsense reasoning aims to address the limitations of purely numerical systems, by providing coarser and more intuitive knowledge representation and reasoning techniques (Kuipers, 1994). The subdiscipline of qualitative spatial and temporal reasoning (QSTR) aims to formalise our intuitive understanding of everyday physical relationships such as size, orientation, topology, and distance, and temporal relation-ships such as ordering, coincidence, and duration (Cohn and Renz, 2007). A seminal example of QSTR is Allen's interval calculus (Allen, 1983) which defines a set of thirteen atomic relations between time intervals, and an algorithm for reasoning about networks of temporal relations, e.g. (where • is a composition operator)

t_1 before t_2 • t_2 contains t_3 = t_1 before t_3
t_1 overlaps t_2 • t_2 during t_3 =
t_1 (overlaps, or during, or starts) t_3

Despite significant theoretical advances, there is a distinct absence of applications that make use of these techniques (Dylla et al., 2006). The central problem is the significant lack of support for adapting and integrating existing QSTR methods with other domain specific qualitative spatial and temporal models. Specifically, there are no methods for assessing the quality of a QSTR system or comparing QSTR systems to determine which one is the most effective for solving a given problem.

Most of the existing research on QSTR analysis has focused on proving correctness of composition (Wölfl et al., 2007), characterising reasoning complexity (Vilain et al., 1989) and determining tractable subsets of well known formalisms (Nebel and Bürckert, 1995). While it is important to know when a problem is NP complete, reasoning complexity cannot be used by the developer to determine whether the particular QSTR system addresses the task at hand, how a change to the system will impact its effectiveness, or how QSTR application validation can be performed to ensure that the application is fit for purpose.

To address this we present a fundamental metric called *H*-complexity that quantifies the expressiveness of a given QSTR system, in terms of the number of distinct scenario classes that can be encoded by the system. The developer can use expressiveness to guide application design in a range of powerful ways, for example:

(i) comparing different QSTR systems to help determine whether one is more suitable than the other;
(ii) calculating test coverage during the validation phase of development to quantify confidence in the application being fit for purpose;

(iii) quantifying the reduction in expressiveness due to constraints to assist in developing a test plan.

In this paper we derive a very simple equation for calculating *H*-complexity, and we consider two applications addressing points (ii) and (iii) above, where we derive very simple equations and a reference table for improved runtime efficiency. We thereby demonstrate that our complexity metric is a flexible and effective tool for supporting the development of QSTR applications.

The remainder of the paper is structured as follows. In the following section we review the basic theory of QSTR systems. We then derive *H*-complexity in two steps. Firstly we define the concept of homogeneous sets as the fundamental folding of objects into equivalence classes permitted by the QSTR language. We then use homogeneous sets to quantify QSTR expressiveness by identifying a one-to-one relationship between accessible, unique subsets and scenario classes. In the succeeding section we derive methods for calculating homogeneous sets in relations of any arity. We then present two methods for employing *H*-complexity: calculating test coverage for validation, and quantifying the reduction in expressiveness due to constraints. In the final section we present the conclusions of this paper.

Foundations of QSTR

Informally, QSTR applications model, infer, and check the consistency of object relations in a scenario. We will define QSTR applications in terms of model theory (Marker, 2002; Hodges, 1997) and then define the roles of QSTR application designs and users.

We use the notation $\uparrow$ to represent the exponent operator, $x{\uparrow}y=x^y$. In model theoretic terms, a language L (or *vocabulary*, or *signature*) is a finite set of relation symbols $\mathbf{R}$ and arities a_R for each $R\in\mathbf{R}$. A model $\mathbf{M}$ of language L (or *L-structure*, or *interpretation*) consists of a *universe* U (or *domain*, or *underlying set*) and for each relation symbol $R\in\mathbf{R}$ there is a set $R_\mathbf{M}\subseteq U\uparrow a_R$. That is, $\mathbf{M}$ provides a concrete interpretation of the symbols in L based on the underlying set U. Finally, a *scenario* (or *configuration*, or *substructure*) is a model $\mathbf{V}$ that can be *embedded* into $\mathbf{M}$, that is, an injective homomorphism $f{:}V\rightarrow U$ exists such that, for each $R\in\mathbf{R}$ with arity a,

$$\forall\ v_1,\ldots,v_a\in V^a\cdot(v_1,\ldots,v_a)\in R_\mathbf{V}\leftrightarrow(f(v_1),\ldots,f(v_a))\in R_\mathbf{M}.$$

A QSTR application has a language L that specifies the set of relation symbols that the designer has deemed relevant to the task at hand. The model $\mathbf{M}$ of a QSTR application is the interpretation of the relations, implemented using *constraints* between the relations (what objects must, or must not, exist in different combinations of relations). For each relation type $R\in\mathbf{R}$ with arity a_R, and for each tuple of arity a_R, the relation either *holds*, *does not hold*, or is *not applicable* for that tuple. Thus, for each relation symbol $R\in\mathbf{R}$ in the language, a QSTR application model $\mathbf{M}$ requires three sets, $R_\mathbf{M}^+$ (holds), $R_\mathbf{M}^-$ (does not hold) and $R_\mathbf{M}^\sim$ (not applicable), with the axiom

Axiom 1. $\forall\ R\in\mathbf{R}\cdot U\uparrow a_R=R_\mathbf{M}^+\ \Delta\ R_\mathbf{M}^-\ \Delta\ R_\mathbf{M}^\sim$,

where Δ is symmetric difference (the set theoretic equivalent of mutual exclusion). For brevity we will omit the $_\mathbf{M}$ and simply write R^+.

The application *designer* is responsible for selecting an appropriate set of relation symbols and encoding an appropriate set of constraints. A QSTR application *user* can then construct scenarios by specifying a model $\mathbf{V}$ and reasoning is used to determine whether the scenario is valid with respect to the model $\mathbf{M}$ (by proving that $\mathbf{V}$ can be embedded in $\mathbf{M}$). Table 1 relates these roles to the formal semantics in model theory and QSTR applications.

Often parts of the user's scenario are indefinite or unknown, and reasoning with the application constraints is used to help resolve this ambiguity. For each relation $R\in\mathbf{R}$, the user can place tuples (of objects from V) with arity a_R in a fourth *indefinite* set, $R_\mathbf{M}^?$ that is mutually exclusive with the three corresponding *definite* sets. This is a shorthand for specifying a set of models $\mathbf{V}_1,\ldots,\ \mathbf{V}_n$ each representing a *possible* scenario.

An example of a scenario is

V={*kitchen*, *lounge*, *study*}
adjacent$^+$ ={(*lounge*, *study*), (*lounge*, *kitchen*)}
adjacent$^?$ ={(*lounge*, *lounge*), (*study*, *lounge*), …}
adjacent$^-$ ={}.

The *adjacent* relation can be defined as symmetric using the constraint $\{(x,y)\mid(y,x)\in adjacent^+\}\subset adjacent^+$. The *LHS* of the constraint as evaluated in the scenario is {(*study*,*lounge*), (*kitchen*,*lounge*)}. The *RHS* as evaluated in the scenario does not contain these tuples as required by the proper subset relation, and so reasoning moves the offending tuples out of *adjacent*$^?$ and into *adjacent*$^+$ thus satisfying symmetry.

Model Theory	*QSTR Application Domain*	*Actor*
language L	specification of useful qualitative relations	QSTR application *designer*
model $\mathbf{M}$ based on L	constraints that determine the interaction between the relations	
model $\mathbf{V}$ based on L embedded in $\mathbf{M}$	Using a QSTR application to represent and reason about objects.	QSTR application *user*

Table 1. Comparing the domains of model theory, QSTR applications, and the roles of QSTR application designers and users.

Homogeneous and Definable sets

In model theory (Marker, 2002), a set X is *definable* in model $\mathbf{M}$ if there is some formula ϕ such that $X=\{(v_1,\ldots,v_n)\in U^n\mid\mathbf{M}\vDash\phi(v_1,\ldots,v_n)\}$ (where entails $\vDash$ means that the formula is true in $\mathbf{M}$). Alternatively, if no formula exists that can separate two objects, then the objects are considered equivalent, and we say that they are in the same *homogeneous* set. Let $H=\{h_1,\ \ldots,\ h_n\}$ be a set of homogeneous sets, where each $h_i \subseteq U$. By definition, $h_1, \ldots, h_n$ partition U, that is, they are mutually exclusive and jointly exhaustive. A homogeneous set h is *evaluated* in scenario s, $s(h) \subseteq V$.

All possible queries are equivalent to some union of homogeneous sets. We define a query q to be a set of homogeneous sets $q=\{h_x,\ldots,h_y\}$, and we say a query q is executed in scenario s, $s(q)= s(h_x)\cup\ldots\cup s(h_y)$.

Constraints

The model of a QSTR application is defined by constraints between the qualitative relations. If a scenario does not satisfy a constraint then reasoning attempts to move the offending tuples out of indefinite sets ($R^?$) and into one of the definite sets (R^+, R^-, or $R^{\sim}$). If offending tuples are not indefinite then reasoning has identified a contradiction and the scenario is inconsistent.

Let constraint $c = X\ \delta\ Y$, where $\delta \in \{\subset,\subseteq,\not\subset,\neq,=,\ldots\}$ is a set comparison and X, Y are set expressions that are evaluated in scenarios. Set expressions are either sets, or the result of set operations.

We now define the complete set of possible comparisons from which δ can be selected. Two sets *LHS*, *RHS* either share some objects, or they do not. The degree of overlap can be used to define all possible set relationships. Let $LHS = h_w\cup\ldots\cup h_x$ and $RHS = h_y\cup\ldots\cup h_z$, let $q_{LHS}=\{h_x,\ldots,h_y\}$ and $q_{RHS}=\{h_x,\ldots,h_y\}$ (queries for *LHS* and *RHS* respectively). Let

$q_L= q_{LHS} / q_{RHS}$	(H sets exclusively in *LHS*)
$q_R= q_{RHS} / q_{LHS}$	(H sets exclusively in *RHS*)
$q_{LR}= q_{LHS} \cap q_{RHS}$	(H sets in both *LHS* and *RHS*)

For any pair of sets *LHS* and *RHS*, the queries q_L, q_R, and q_{LR} will evaluate to be either empty or non-empty, giving $2^3=8$ different basic set comparisons, as illustrated in Table 2. In each table entry, the outer circle represents the set of all scenarios, and three inner circles each represent a subset of scenarios where a condition holds. The top left circle specifies those scenarios where H sets in the query q_L are empty, $s(q_L)=\varnothing$ and so on. For example, the constraint $X = Y$ is satisfied in exactly those scenarios where both q_L (elements exclusively in X) and q_R (elements exclusively in Y) evaluate to empty and q_{LR} (elements shared by both X and Y) is not empty. The comparison δ can consist of any disjunction of these basic set comparisons (e.g. $\subseteq$ is $= \vee \subset$), thus there are $2^8-1 = 255$ possible comparisons.

Basic Combinatorics

Finally, we provide some basic equations from combinatorics that will be used in the paper. The size of the powerset of set X is $2^{|X|}$. The number of subsets that contain elements from either set X or set Y is $2^{|X \cup Y|}$, the number of subsets that contain all elements from X and not Y is $2^{|X\ /\ Y|}$, and the number of subsets that contain all elements from X and neither Y nor Z is $2^{|X / (Y \cup Z)|}$.

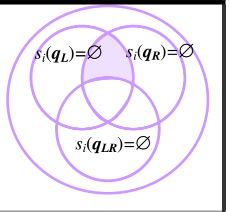
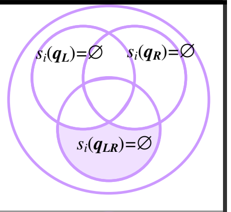
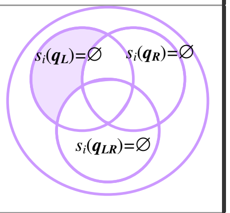
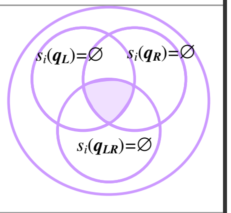
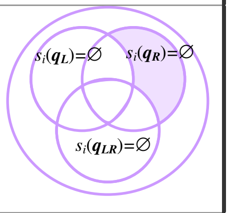
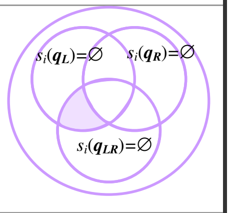
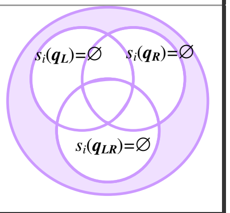
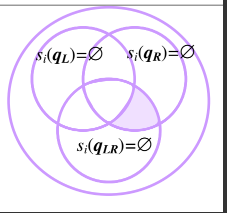
δ	*scenarios (shaded=valid scenarios)*	δ	*scenarios (shaded=valid scenarios)*
$=$	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$	∩̶	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$
$\subset$	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$	$=_\varnothing$	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$
$\supset$	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$	$\subset_\varnothing$	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$
${}^{+}\cap^{\neq}$	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$	$\supset_\varnothing$	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$

Table 2. Eight basic relationships between sets derived from the overlap between H sets. Outer circles represent the set of all scenarios, and three inner circles each represent a subset of scenarios where a condition holds. Shaded regions represent scenarios that satisfy the set relationship.

Equivalently, we use bit-string encodings, where the number of unique combinations that can be represented by a bit-string of length n is 2^n. The number of combinations of two bit-strings of length n_1 and n_2 is 2^{n1+n2}. If m bits are fixed in a bit-string of length n then the number of unique combinations that contain the m fixed bits is 2^{n-m}.

Table 3 gives some equations for calculating the number of scenarios that satisfy conditions required by constraints. For example, in each table entry the top left circle specifies those scenarios where H sets in the query q_L are empty, $s(q_L)=\varnothing$, and the number of such scenarios is $2^{|H\ /\ qL|} = 2^{|H| - |qL|}$. To understand this, consider a bit-string of length $n=|H|$, where $m=|q_L|$ bits are fixed to '0' (representing that $s(q_L)=\varnothing$); the number of unique combinations that this bit-string allows is $2^{n-m} = 2^{|H|-|qL|}$. Scenarios where H sets in both q_L and q_R are empty, $s(q_L)=\varnothing \wedge s(q_R)=\varnothing$ is $2^{|H / (qL \cup qR)|} = 2^{|H|-|qL|-|qR|}$ (recall that, by definition, q_L and q_R are mutually exclusive). A bit-string of length $n=|H|$, where $m=|q_L \cup q_R|=|q_L|-|q_R|$, bits are fixed to '0' admits $2^{n-m} = 2^{|H|-|qL|-|qR|}$ unique combinations.

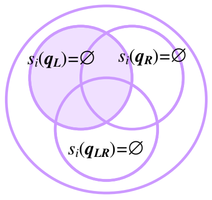
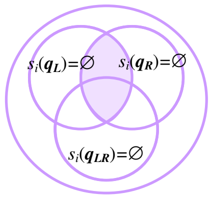
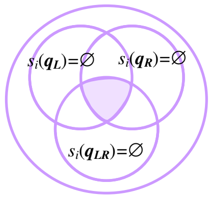
formula	*scenarios (shaded=valid scenarios)*
$2^{\lvert H\rvert-\lvert qL\rvert}$ *symmetric with* $2^{\lvert H\rvert-\lvert qR\rvert}$ $2^{\lvert H\rvert-\lvert qLR\rvert}$	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$
$2^{\lvert H\rvert-\lvert qL \cup qR\rvert}$ *symmetric with* $2^{\lvert H\rvert-\lvert qL \cup qLR\rvert}$ $2^{\lvert H\rvert-\lvert qLR \cup qR\rvert}$	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$
$2^{\lvert H\rvert-\lvert qL \cup qR \cup qLR\rvert}$	 $s_i(q_L)=\varnothing$ $s_i(q_R)=\varnothing$ $s_i(q_{LR})=\varnothing$

Table 3. Combinatorial formulae for calculating the number of scenario classes that satisfy particular conditions.

H-Complexity

In this section we present the central theory of *H*-complexity. *H*-complexity is a measure of the number of different scenario classes that can be encoded by a given QSTR language. It identifies the most extreme partitioning of objects that the language allows, so that all other practical partitioning schemes will consist of some subset of the distinctions made in *H*-complexity.

Homogeneous Sets

If no possible set theoretic query exists that can separate two objects, then the objects are considered equivalent. This inherent limitation fundamentally folds objects into homogeneous sets, or *H* sets, providing the foundation for a measure of expressiveness or complexity. *H* sets represent the maximum refinement permitted by the QSTR language, i.e. the point where no further distinctions are possible. Thus, *H*-complexity = $|H|$.

Assume for simplicity that all relations have an arity of 1 (i.e. they represent qualitative properties such as *round* or *large*). If there are n relations, and each relation is represented by four sets, then there are 4^n different combinations of these relations, i.e.

1. $R_1^+ \cap R_2^+ \cap \ldots \cap R_n^+$,
2. $R_1^- \cap R_2^+ \cap \ldots \cap R_n^+$,

…

4^n. $R_1^{\sim} \cap R_2^{\sim} \cap \ldots \cap R_n^{\sim}$.

In general,

$$|H|=\prod_{R\in \mathbf{R}} |H_R| \qquad (1)$$

where $|H_R|=|A_R|$ for unary relations, A_R is the selection of relation sets used by R (e.g. + and −), and $\prod$ is the sequence product operator. We calculate *H* sets for relations with higher arities in a later section.

For *H* sets to truly represent the maximum refinement possible, they must be jointly exhaustive and pairwise disjoint (JEPD) so that every object in a scenario will appear in exactly one *H* set. This property is critical; if it did not hold then further refinements could be achieved by taking *H* set intersections and differences. The *H* sets from Equation 1 are pairwise disjoint because, for any relation R_i, the relation's four sets are mutually exclusive, and any two *H* sets always differ by at least one R_i set. They are also jointly exhaustive because, for any relation R_i, each of its four jointly exhaustive sets is covered by some *H* set.

Query Complexity

A query is used to access a subset of objects in a scenario. Query complexity is the maximum number of unique non-empty subsets that can be accessed by some query. We will now show that all accessible, unique subsets of objects in a scenario can be represented as the union of some combination of *H* sets.

H sets are indivisible and mutually exclusive, and so the query that returns the smallest non-empty subset of objects contained in an *H* set is the set expression of the *H* set itself, e.g. $R_1^+ \cap R_2^+ \cap \ldots \cap R_n^+$. The smallest subset containing objects from two different *H* sets h_1, h_2 is the union of those two *H* set expressions $h_1 \cup h_2$. It follows that any accessible subset of objects must be the union of some combination of *H* sets. Thus query complexity is the number of different combinations of *H* sets, $2^{|H|}$.

Scenario Complexity

If two objects in a scenario can not be separated by a query, then the objects are considered equivalent and indistinguishable, that is, the objects are in the same *H* set. If the only difference between two scenarios is the number of indistinguishable objects in each non-empty *H* set, then the scenarios are considered equivalent. This follows the intutive understanding that qualitative models, unlike metric systems, do not deal with numerical quantities. Thus, a scenario equivalence class is defined by the combination of *H* sets that are empty and non-empty, $2^{|H|}$.

There is an interesting parallel between query and scenario complexity. The number of unique, accessible, non-empty subsets of a QSTR application is equal to the number of distinct scenario classes that can be expressed.

An alternative interpretation is that a scenario is defined by the combination of queries that return non-empty results, i.e. if query q_1 returns $\varnothing$ in scenario s_1 and some non-empty result in s_2, then the scenarios are logically distinct.

Calculating Homogeneous Sets

In this section we derive the equation for $|H_R|$ admitted by a relation R of arity a_R, and then derive $|H|$ for a QSTR system. Once relations have an arity greater than 1 there are an infinite number of potential H sets, because a binary relation constitutes a total order. Thus, Equation 1 cannot be used to compare QSTR systems with binary or higher relations. We proceed in our analysis by representing a scenario of binary relation R_i as a graph, where objects represent vertices and directed edges represent tuples as illustrated in Figure 1.

$R_1=\{ (a,b), (c,b)\}$

$R_2=\{(a,a), (a,b), (a,c), (d,d), (d,e), (e,c)\}$

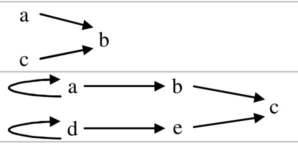


Figure 1. Two graphs representing binary relations R_1 and R_2.

A set theoretic query describes the structure of a graph and specifies the vertex to be selected with v bound variables,

$\exists x_1 ... \exists x_v \cdot x_1 \neq x_2 \wedge ... \wedge x_1 \neq x_v \wedge \quad ... \wedge x_{v-1} \neq x_v$.

For brevity, we will omit explicitly stating these quantifications and conditions for all further queries, and for simplicity we do not allow the universal quantifier $\forall$. For example, the query $\{x_2| (x_1,x_2)\in r_1 \wedge (x_3,x_2)\in R_1\}$ will access b from the graph of R_1 in Figure 1.

Calculating $|H_R|$ with Arbitrary Arity

While there are an infinite number of potential graphs and unique accessible subsets, homogeneous sets still exist that contain indistinguishable objects. For example, regarding the graph of R_1, no query exists that can separate objects a and c (without directly referring to those objects),

$\{a,c\}=\{x_1| (x_1,x_2)\in R_1 \wedge (x_3,x_2)\in R_1\}$
$=\{x_3| (x_1,x_2)\in R_1 \wedge (x_3,x_2)\in R_1\}$,

and the graph of R_2 has three H sets, accessed by the query

$\{x_i \mid (x_1,x_1)\in R_2 \wedge (x_1,x_2)\in R_2 \wedge (x_2,x_3)\in R_2 \wedge (x_4,x_4)\in R_2 \wedge (x_4,x_5)\in R_2 \wedge (x_5,x_3)\in R_2 \}$,

namely $\{a,d\}$ when i=1 or 4, $\{b,e\}$ when i=2 or 5 and $\{c\}$ when i=3. Thus, homogeneous sets correspond to graph symmetries or automorphisms. Given a graph of a scenario, the number of H sets is the number of vertices minus the number of automorphisms.

In order to make $|H_r|$ a function of QSTR systems rather than scenarios (particular graphs), the query language must be restricted. If the restricted language only recognises a finite number of graphs, it will admit a finite number of H sets. It is then possible to quantify the complexity of a relation independent of a particular scenario, and measure the relative difference in expressiveness between two QSTR systems. Two basic restrictions are to limit the number of variables (vertices) and the number of tuples (edges). We will consider the former case. Previously, queries have referred to variables x_i where i can be any positive integer. The strongest restriction on the number of variables is $i\leq 1$, affording only one query, $\{x_1| (x_1,x_1)\in R_i\}$. If $i\leq 2$ then the allowable tuples are (x_1,x_1), (x_1,x_2), (x_2,x_1), and (x_2,x_2). If v is the number of variables allowed in a query, and a_R is the arity of relation R (i.e. the size of the tuples) then for each query,

$$\text{number of tuples} = v\uparrow a_R$$

All binary tuples of n objects in relation R_i appear in exactly one of the R_i sets (holds etc.), and thus we focus on those queries that contain all $v\uparrow a_R$ tuples permitted by the language. We refer to these as atomic queries. The difference between two atomic queries is the combination of R_i sets in which the tuples must appear. For example, Table 4 gives an extract of the atomic queries where v=2 and a_R=2. Some graphs are automorphic, e.g. in row 1, and some graphs are isomorphic, e.g. rows 2 and 3. Initially there are 16 graphs × 2 variables = 32 different executable atomic queries. 4 graphs (8 queries) have pairs of automorphic variables making 8/2 queries redundant, and the remaining 12 graphs (24 queries) can be put into symmetric pairs, making an additional 24/2 queries redundant, leaving 32 – 4 – 12 = 16 unique atomic queries. An easy way to arrive at the number of unique atomic queries is to allow every graph, but restrict the selection to the first variable x_1 (this can be viewed as fixing the variable and rotating the edges of the graph to cover symmetries). Thus,

$$\text{number of unique atomic queries} = |A_R|^{\text{number of tuples}}$$

where *number of tuples* = $v\uparrow a_R$. Finally, to calculate $|H_R|$ we must determine the smallest JEPD queries that contain the atomic queries. Atomic queries are not JEPD when their corresponding graphs are overlapping induced subgraphs of the full scenario graph. For example, consider the scenario graph in Figure 2.

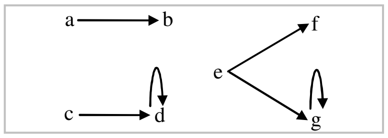


Figure 2. Graph consisting of three disconnected subgraphs. Subgraphs $\{a,b\}$ and $\{c,d\}$ correspond to atomic queries, where $\{e,f,g\}$ contains both atomic queries as induced subgraphs.

If v=2 then two atomic queries are:

$\{x_1 \mid (x_1,x_1)\in R^- \wedge (x_1,x_2)\in R^+ \wedge (x_2,x_1)\in R^- \wedge (x_2,x_2)\in R^-\}=\{a,e\}$,

$\{x_1 \mid (x_1,x_1)\in R^- \wedge (x_1,x_2)\in R^+ \wedge (x_2,x_1)\in R^- \wedge (x_2,x_2)\in R^+\}=\{c,e\}$.

The atomic queries are not JEPD, as vertex e appears in both results. However, if we take all combinations of atomic queries by intersection and difference, we can produce a JEPD collection of H sets, hence

$$|H_R|=2^{\text{number of unique atomic queries}} - 1$$

where the *number of graphs* = $|A_R|^{\text{number of tuples}}$. To summarise,

$$|H_R|=(2\uparrow|A_R|\uparrow v\uparrow a_R) - 1. \quad (2)$$

Calculating $|H|$ with Arbitrary Arity

The problem is now calculating $|H|$, the total number of H sets across all relations when those relations can take any arity. Previously, we only referred to one relation within a query. Given v bound variables, queries will now take the form,

$\{x_1 \mid \quad \textit{query } R_1, \textit{query } R_2, \ldots, \textit{ query } R_n\}$

where *query* R_i is one of the unique atomic queries for relation R_i. The number of queries permitted in this form is

$|\textit{atomic } R_1 \textit{ queries}| \times \ldots \times |\textit{atomic } R_n \textit{ queries}|$.

We have shown that the number of atomic queries for R_i is $|A_{ri}|\uparrow v\uparrow a_{ri}$, thus

$$\begin{aligned}|H| &= 2\uparrow(\ |\textit{atomic } R_1 \textit{ queries}| \\ &\qquad \times\ldots\times \\ &\qquad |\textit{atomic } R_n \textit{ queries}|) \\ &= 2\uparrow((|A_{R1}|\uparrow v\uparrow a_{R1}) \times\ldots\times (|A_{R\,n}|\uparrow v\uparrow a_{R\,n})) \\ &= 2\uparrow \textstyle\prod_{R\in \mathbf{R}} (|A_R|\uparrow v\uparrow a_R). \quad (3)\end{aligned}$$

Using this formulation we can identify some basic properties of expressiveness. In general, H-complexity is a function of the number of atomic queries that the QSTR language allows. Restricting the number of variables in a query to v makes complexity a function of the number of relation states, $|A_R|$, the number of variables v, and the relation arity. Moreover, Equation 3 specifies the relative influence that each component has on complexity; relation arity has the most influence, followed by the number of variables having exponentially less influence, and finally arity having exponentially less influence again. These properties help to inform the developer about how changes to each component affects expressiveness, and the relative difference in expressiveness between two QSTR systems.

Applying H-Complexity

In this section we present two ways that H-complexity can be employed to assist QSTR application development.

Test coverage

Application test space is defined according to system inputs and outputs, and the system structure such as decisions and control paths. Covering the entire, often infinite test space is clearly impractical and thus software engineers employ methods that isolate key subsets such as boundary checking, equivalence class partitioning, and cause-effect graphs (Burnstein, 2003). Test coverage is a standard metric in software engineering used to guide testing. H-complexity can be used to quantify a QSTR application's potential test space for calculating test coverage, as it specifies the degree of refinement permitted by the language. Consider the following application-specific constraint for determining apparent room colour temperature (Schultz et al., 2009):

"If a room has at least one warm light, and does not have lights of any other temperature, then the room has a warm colour temperature"

This constraint may then be integrated into an existing QSTR system that reasons about spatial arrangements of light sources and surfaces. One formal encoding of this constraint might be

$$\{x \mid (\exists y.\ light^+(y) \wedge in^+(y,x) \wedge warm^+(y)) \quad (3)$$
$$\wedge\neg(\exists y'.\ light^+(y') \wedge in^+(y',x) \wedge warm^-(y'))\} \subseteq warm^+.$$

Based on the structure of the constraint, the developer may decide to test the class of scenarios where the LHS is not empty $\exists x \cdot \exists y\wedge\neg\exists y'$, and where the LHS is empty $\forall x \cdot \exists y\wedge\exists y'$. Based on the relations in the constraint, the developer may decide to test all combinations of conditions for both y and y'; noting that not all conditions are independent (e.g. if there exists y such that $y\in warm^+$ then there must also exist y' such that $y'\in warm^+$), this provides a test set size of 2^8. The issue is that the developer needs to know how many tests must be executed before achieving some level of confidence that the application is fit for purpose. Three test coverage metrics are

1. proportion of H sets tested, $|H_T| / |H|$
2. proportion of relevant H sets tested, $|H_T\cap H_\Delta| / |H_\Delta|$
3. degree of tested combinations for particular clusters of H sets (e.g. some subset $H'\subset H$ tested up to all triples)

where the H_T is the set of tested homogeneous sets, and relevant homogeneous sets H_Δ are defined according to the application (e.g. using probability distributions over inputs, and weighting critical inputs that must be handled correctly). If H_T is not precisely known, then it can be roughly approximated if the tradeoff between the number of tested H sets and the degree of combination testing is known[2]:

a) exhaustive combination testing, $\log_2(|T|)\approx|H_T|$,
b) k combinations, $\lfloor(k!\cdot|T|)^{1/k} + k/2\rfloor \approx|H_T|$, and
c) up to k combinations, $\lfloor(k!\cdot|T|)^{1/k}\rfloor \approx|H_T|$.

Continuing with the case study, given A={+,−}, v=2 and a=2, the *in* relation yields $|H_{in}|=2^{16}-1$, and together *light* and *warm* yield $|H_{warm}|\cdot|H_{light}|=2^2$. Thus, the potential test space is intractable, i.e. applying Equation 2 gives

[2] Formulae are derived by rearranging the standard combination formula, ${}_nC_k=n! / (k!(n-k)!)$ where $n=|T|$ and identifying the dominate terms.

$2\uparrow(2^{18}-2^2)$ containing $\sim 2.6\cdot 10^5$ types of distinction. However, this space includes an exhaustive enumeration of tests that violate basic properties. We can therefore focus the potential test space, with respect to the rule at hand, by incorporating the following restrictions:

- *in* is neither reflexive nor symmetric
- no object is ever *in* a *light*

Rather than exhaustively testing these basic constraints in every rule, the developer may exercise those properties in isolation (e.g. testing when in^+ erroneously contains a pair of symmetric tuples) and then assume they hold when testing other rules. The first constraint allows three queries,

$q_1=\{x \mid (x,y)\in in^+ \wedge (y,x)\in in^-\}$,
$q_2=\{x \mid (x,y)\in in^- \wedge (y,x)\in in^+\}$, and
$q_3=\{x \mid (x,y)\in in^- \wedge (y,x)\in in^-\}$,

giving $|H_{in}|=2^3-1$. The second constraint states that for all scenarios, $q_2 \cap light^+=\varnothing$, hence H sets that contain this query can be removed. 2^{3-1} sets in H_{in} intersect with q_2 giving $|H_{in \text{ with } q2}|\cdot|H_{light+}|\cdot|H_{warm}| = 2^2\cdot 1\cdot 2 = 8$. Therefore, the relevant number of homogeneous sets is $|H_\Delta| = |H_{in}|\cdot|H_{light}|\cdot|H_{warm}| - 8 = (2^3-1)\cdot 2\cdot 2 - 8 = 20$. Recall that in our example above, the developer has chosen exhaustive combination testing of selected relevant components $|T|=2^8$, giving $|H_T|\approx\log_2(|T|)=8$. Test coverage results are

1. $|H_T|/|H| = 8/(2.6\cdot 10^5) \approx 0\%$
2. $|H_T \cap H_\Delta| / |H_\Delta| = 8/20 = 40\%$
3. all combinations $k=1\ldots|H_T|$ of the cluster H_T are tested, and no other clusters are tested at all.

Firstly note that a few key restrictions (particularly the number of tuples in a query) rapidly focus the test space. Secondly it is clear that the majority of the relevant test space is completely untested. The developer can now identify the untested distinctions and decide whether further tests are required.

Constraints and Expressiveness

As we observed in the previous section, a designer can isolate scenario classes by encoding constraints. This is an effective method for improving testing efficiency by isolating and independently testing particular model fragments (to avoid combinatorial explosion). *H*-complexity can be used to measure the number of scenario classes that satisfy a collection of constraints, thus allowing a designer to quantify the reduction in the test suite.

Consider a simple system with two unary relations, R_1 and R_2, $A=\{+,-\}$, and a constraint $R_1^+\subseteq R_2^+$. From Equation 1 we have $|H|=4$ giving 2^4 unique scenario classes, but not all of them will satisfy the constraint. To determine the set of valid scenario classes, the constraint is translated into a union of H sets.

Let $h_1=(R_1^+\cap R_2^+)$, $h_2=(R_1^+\cap R_2^-)$, $h_3=(R_1^-\cap R_2^+)$, $h_4=(R_1^-\cap R_2^-)$;
$R_1^+= h_1 \cup h_2$
$R_2^+= h_1 \cup h_3$

Therefore,

$R_1^+\subseteq R_2^+ \equiv h_1 \cup h_2 \subseteq h_1 \cup h_3$
$\equiv h_2 \subseteq h_3$ *…remove* h_1 from both sides

So a scenario is only valid if $h_2 \subseteq h_3$, but by definition homogeneous sets are mutually exclusive, $h_2 \cap h_3 = \varnothing$. It follows that the only valid scenarios are those where $h_2 = \varnothing$. Given n constraints of the form $LHS_i \subseteq RHS_i$, where $LHS_i = h_{iw}\cup\ldots\cup h_{ix}$ and $RHS_i = h_{iy}\cup\ldots\cup h_{iz}$, and let $q_{Li}=\{h_{iw},\ldots,h_{ix}\} / \{h_{iy},\ldots,h_{iz}\}$ (H sets exclusively in LHS). The only valid scenarios are those where every LHS_i is empty, $s(q_i)=\varnothing$, thus (refer to the previous section "Basic Combinatorics")

$|valid\ scenarios| = 2\uparrow(|H| - |q_{L1}\cup\ldots\cup q_{Ln}|)$.
% of valid scenarios
$= |valid\ scenarios| / |all\ scenarios|$
$= 2\uparrow(|H| - |q_{L1}\cup\ldots\cup q_{Ln}|) / 2\uparrow|H|$
$= 2\uparrow(-|q_{L1}\cup\ldots\cup q_{Ln}|)$.

We will derive a function $\underline{\boldsymbol{\theta}}_T$ for calculating valid scenarios for any type of constraint. Let reference Table 4 accept a constraint and return the appropriate formula for calculating valid scenarios. Note that each table entry takes the form

$$a_0 2^{-e0} + \ldots + a_n 2^{-en}$$

where each term $j=0\ldots n$ has a sign $a_j\in\{1,-1\}$ and an integer exponent e_j. Let *term map* θ_T be a map that collects the sign and size of the exponent from each term in this series. That is, θ_T takes a constraint c_i and returns a list $\{(a_0, e_0), \ldots, (a_n, e_n)\}$ such that for each tuple (a_j, e_j) there exists some term $\ldots+ a_j 2^{ej} +\ldots$ in the Table 4 entry for c_i.

We can now recursively define the function $\underline{\boldsymbol{\theta}}_T$ that accurately calculates the term information from n constraints, for $i=2\ldots n$,

$$\underline{\boldsymbol{\theta}}_T(c_1) = \{(a, \mathrm{f}(X)) \mid (a, X)\in \theta_T(c_1)\} \quad (3.1)$$
$$\underline{\boldsymbol{\theta}}_T(c_i) = \underline{\boldsymbol{\theta}}_T(c_{i-1}) \times \theta_T(c_i) \quad (3.2)$$

such that, given $(a_x, X)\in \underline{\boldsymbol{\theta}}_T(c_{i-1})$ and $(a_y, Y)\in \theta_T(c_i)$, $(a_x, X)\times(a_y, Y)=(a_x a_y, \mathrm{g}(X, \mathrm{f}(Y)))$. The function $\mathrm{f}(X)$ determines the information about the exponent that is used, and $\mathrm{g}(X, Y)$ determines how the exponent information of different terms should be combined. We evaluate the terms using the formula (where f^{-1} is the inverse of f)

% valid scenarios $= \sum a_j 2\uparrow - |\mathrm{f}^{-1}(e_j)|$.

For example, let two constraints be
$c_1= h_1 \cup h_2 \cup h_3 = h_3 \cup h_4 \cup h_5$
$c_2= h_6 \cup h_7 \subseteq h_1 \cup h_7 \cup h_8$.

$q_{1L} = \{h_1, h_2\}$, $q_{1R} = \{h_4, h_5\}$, $q_{1LR} = \{h_3\}$
$q_{2L} = \{h_6\}$, $q_{2R} = \{h_1, h_8\}$, $q_{2LR} = \{h_7\}$

$\theta_T(c_1) = \{ (1, \{h_1, h_2, h_4, h_5\}),$
$(-1, \{h_1, h_2, h_3, h_4, h_5\}) \}$

$$
\begin{aligned}
\theta_T(c_2) = \{\quad & (\ \ 1, \{h_6\}),\\
& (-1, \{h_1, h_6, h_8\})\\
& (-1, \{h_6, h_7\})\\
& (\ \ 1, \{h_1, h_6, h_7, h_8\}), \qquad \textit{...for} \subset\\
\\
& (\ \ 1, \{h_1, h_6, h_8\}),\\
& (-1, \{h_1, h_6, h_7, h_8\}) \qquad \textit{...for} =\\
\}
\end{aligned}
$$

Let f(X)=X and g(X, Y)=$X \cup Y$ (see Table 5).

$$
\begin{aligned}
\underline{\boldsymbol{\theta}}_T(c_1) &= \theta_T(c_1)\\
\underline{\boldsymbol{\theta}}_T(c_2) &= \underline{\boldsymbol{\theta}}_T(c_2) \times \theta_T(c_1)\\
&= \{(1, \{h_6\}) \times (\ \ 1, \{h_1, h_2, h_4, h_5\}),\\
&\quad (1, \{h_6\}) \times (-1, \{h_1, h_2, h_3, h_4, h_5\}),\\
&\quad \ldots,\\
&\quad (-1, \{h_1, h_6, h_7, h_8\}) \ \times\\
&\qquad (-1, \{h_1, h_2, h_3, h_4, h_5\}),\\
&= \{(\ \ 1, \{h_1, h_2, h_4, h_5, h_6\}),\\
&\quad (-1, \{h_1, h_2, h_3, h_4, h_5, h_6\}),\\
&\quad \ldots,\\
&\quad (\ \ 1, \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8\})\ \}.
\end{aligned}
$$

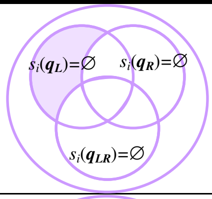
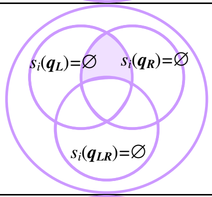
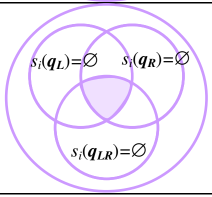
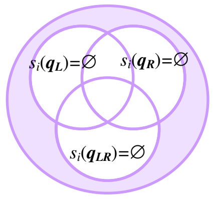
set comparison	*scenarios (shaded=valid scenarios)*	*formula*
$\subset$ *symmetric with* $\supset$ ⩀	 $s_i(\boldsymbol{q_L})=\varnothing$ $s_i(\boldsymbol{q_R})=\varnothing$ $s_i(\boldsymbol{q_{LR}})=\varnothing$	$2^{-\lvert qL\rvert}$ $-\ 2^{-\lvert qL \cup qR\rvert}$ $-\ 2^{-\lvert qL \cup qLR\rvert}$ $+\ 2^{-\lvert qL \cup qR \cup qLR\rvert}$
$=$ *symmetric with* $\supset_\varnothing$ $\subset_\varnothing$	 $s_i(\boldsymbol{q_L})=\varnothing$ $s_i(\boldsymbol{q_R})=\varnothing$ $s_i(\boldsymbol{q_{LR}})=\varnothing$	$2^{-\lvert qL \cup qR\rvert}$ $-\ 2^{-\lvert qL \cup qR \cup qLR\rvert}$
$=_\varnothing$	 $s_i(\boldsymbol{q_L})=\varnothing$ $s_i(\boldsymbol{q_R})=\varnothing$ $s_i(\boldsymbol{q_{LR}})=\varnothing$	$2^{-\lvert qL \cup qR \cup qLR\rvert}$
${}^{+}\cap^{\neq}$	 $s_i(\boldsymbol{q_L})=\varnothing$ $s_i(\boldsymbol{q_R})=\varnothing$ $s_i(\boldsymbol{q_{LR}})=\varnothing$	1 $-\ 2^{-\lvert qL\rvert}$ $-\ 2^{-\lvert qR\rvert}$ $-\ 2^{-\lvert qLR\rvert}$ $+\ 2^{-\lvert qL \cup qR\rvert}$ $+\ 2^{-\lvert qL \cup qLR\rvert}$ $+\ 2^{-\lvert qR \cup qLR\rvert}$ $-\ 2^{-\lvert qL \cup qR \cup qLR\rvert}$

Table 4. Formulae for calculating the number of scenario classes that satisfy particular set comparisons used in constraints.

To avoid actually identifying and testing H sets (which is an extremely resource intensive process due to their enormous quantity) we can calculate bounds by assuming maximum and minimum overlap between the constraint queries. Lower and upper bounds are calculated by changing the definition of the function f, summarised in Table 5.

result	f(X) =	g(X, Y) =	*error tolerance* (ε)
exact	X	$X \cup Y$	$\lvert$g(X, Y)$\rvert \le \alpha$
lower bound	$\lvert X\rvert$	$X + Y$	g(X, Y) $\le \alpha$
upper bound		*max*(X, Y)	

Table 5. Alternative functions used in equations 3.1 and 3.2 for calculating exact, lower and upper bounds of the number of scenario classes that satisfy a set of constraints.

The lower bound of % valid scenarios occurs when no constraints share any H sets, and is calculated by summing exponents g(X, Y) = $X + Y$. The upper bound occurs when every query is an improper subset of some query, and is calculated by taking the maximum exponent, g(X, Y) = *max*(X, Y).

Finally, we can vastly improve performance by pruning a term once its exponent has exceeded some threshold, α. As the exponents grow, the size of the term quickly becomes negligible, i.e. $\lim_{e\to\infty} 2^{-e} = 0$. The threshold α is a function of the required error tolerance ε and the number of terms $m = \sum_{i=0\ldots n} \lvert\theta_T(c_i)\rvert$,

$$\alpha = \lceil \log_2 (m\,/\,\varepsilon) \rceil,$$

derived as follows. The sum of all removed terms (where the magnitude of the exponent exceeds some threshold α) must be less than or equal to the given error tolerance,

$$
\begin{aligned}
\varepsilon &\ge m\, 2^{-\alpha}\\
\varepsilon\,/\,m &\ge 2^{-\alpha}\\
\varepsilon.2^{\alpha}/\,m &\ge 1\\
2^{\alpha} &\ge m\,/\,\varepsilon\\
\alpha &\ge \log_2 (m\,/\,\varepsilon)
\end{aligned}
$$

For example, an application has 50 constraints and we want the error to be within 0.01%. The number of terms m depends on the constraint relations, but for this example on average let each constraint have 6 terms, giving $m = 50 \times 6 = 300$ terms. Therefore,

$$
\begin{aligned}
\alpha &= \lceil \log_2 (300\,/\,0.01) \rceil\\
&= \lceil \log_2 (30000) \rceil\\
\alpha &= 15
\end{aligned}
$$

Conclusions

In this paper we presented the H-complexity metric for analysing QSTR systems, with the aim of supporting the design and evaluation of QSTR applications. H-complexity measures the expressiveness of a QSTR system according to the number of distinct scenario classes that can be encoded by the system. Specifically, we derived *H*-complexity by firstly defining the concept of homogeneous sets as the fundamental folding of objects into equivalence classes permitted by the QSTR language. We then used homogeneous sets to quantify complexity by identifying a one-to-one relationship between accessible, unique subsets and scenario classes. We have shown that *H*-complexity of a relation can be quantified independently of a particular scenario by applying restrictions to the query language. This enables a developer to measure the relative difference in expressiveness between two QSTR systems that contain relations that admit an infinite number of potential homogeneous sets. Furthermore, we have shown that relation arity and the number of query variables significantly dominate expressiveness. This is useful in determining the relative difference in expressiveness between two QSTR systems, which can be used by a developer to guide QSTR application design. Finally, we presented two examples which illustrate how *H*-complexity can be employed to support QSTR application development, firstly, to calculate the potential test space for determining test coverage, and secondly to quantify the reduction in expressiveness due to constraints. These examples demonstrate that our complexity metric is a versatile and effective tool for supporting the development of QSTR applications.

Acknowledgements

This work has been funded by the Bright Future Top Achiever Doctoral Scholarship (Tertiary Education Commission, New Zealand).

References

Allen, J. F. *Maintaining knowledge about temporal intervals*. Communications of the ACM: 26, 11, pp. 832-843: ACM Press, 1983.

Burnstein I. *Practical software testing: a process oriented approach*. Springer, New York, 2003.

Cohn, A. G., and Renz, J. *Qualitative spatial reasoning*. In van Harmelen F, Lifschitz V, Porter B (eds): Handbook of Knowledge Representation: Elsevier Science, 2007.

Dylla, F., Frommberger, L., Wallgrün, J. O., and Wolter, D. *SparQ: A Toolbox for Qualitative Spatial Representation and Reasoning*. In Proc. Workshop on Qualitative Constraint Calculi: Application and Integration, 2006.

Hodges, W. *A Shorter Model Theory.* Cambridge University Press, 1997.

Kuipers, B. *Qualitative reasoning*. MIT Press, 1994.

Ligozat, G., Mitra, D., and Condotta, J. *Spatial and temporal reasoning: beyond Allen's calculus*. AI Communications 17, 4, pp. 223–233, 2004.

Marker, D. *Model Theory: An Introduction.* Springer-Verlag New York, 2002.

Nebel, B., Bürckert, H.J. *Reasoning about temporal relations: a maximal tractable subclass of Allen's Interval Algebra*, Journal of the ACM, 42, 1, pp.43–66, 1995.

Schultz, C., Amor, R., Lobb, B., Guesgen, H., *Qualitative design support for engineering and architecture*. Advanced Engineering Informatics, Elsevier, 23, 1, pp. 68-80, 2009.

Vilain, M., Kautz, H., van Beek, P. *Constraint propagation algorithms for temporal reasoning: a revised report*, Readings in Qualitative Reasoning about Physical Systems, Morgan Kaufmann, San Francisco, CA, USA, 1989.

Wölfl, S., Mossakowski, T., and Schröder, L. *Qualitative constraint calculi: Heterogeneous verification of composition tables*. In Proc. FLAIRS-07, pp. 665-670. AAAI Press, 2007.