Constraint-Based Conversion of Fiction Text to a Time-Based Graphical Representation

Kevin Glass
Department of Computer Science
Rhodes University
Grahamstown, South Africa
k.glass@ru.ac.za

Shaun Bangay
Department of Computer Science
Rhodes University
Grahamstown, South Africa
s.bangay@ru.ac.za

ABSTRACT

This paper presents a method for converting unrestricted fiction text into a time-based graphical form. Key concepts extracted from the text are used to formulate constraints describing the interaction of entities in a scene. The solution of these constraints over their respective time intervals provides the trajectories for these entities in a graphical representation.

Three types of entity are extracted from fiction books to describe the scene, namely Avatars, Areas and Objects. We present a novel method for modeling the temporal aspect of a fiction story using multiple time-line representations after which the information extracted regarding entities and time-lines is used to formulate constraints. A constraint solving technique based on interval arithmetic is used to ensure that the behaviour of the entities satisfies the constraints over multiple universally quantified time intervals. This approach is demonstrated by finding solutions to multiple time-based constraints, and represents a new contribution to the field of Text-to-Scene conversion. An example of the automatically produced graphical output is provided in support of our constraint-based conversion scheme.

Categories and Subject Descriptors

1.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Language parsing and understanding*; 1.3.7 [**Computer Graphics**]: Three-dimensional Graphics and Realism—*Animation*

General Terms

Text-to-Scene conversion, interval constraint solving

Keywords

Constraint Solving, Interval Arithmetic, Text-to-Scene

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SAICSIT 2007 2 - 3 October 2007, Fish River Sun, Sunshine Coast, South Africa
Copyright 2007 ACM 978-1-59593-775-9/07/0010 ...$5.00.

1. INTRODUCTION

1.1 Problem Statement

We investigate the process of converting fiction text to a corresponding graphical form. In particular, we focus on the previously un-investigated case of:

- unrestricted input text (directly using text written for human consumption), and
- taking the temporal nature of fiction writing into account.

This paper presents a constraint-based approach to addressing this problem by defining the categories of entity required for graphical representation, and creating constraints relating these in a temporal manner. We describe how such constraints are applied to actual object models, and present a solving technique based on interval arithmetic which is capable of ensuring that these non-linear constraints are satisfied over the specified time periods.

1.2 Background

The process described in this paper is a component of a Text-to-Scene conversion system, where information from natural language texts is used to populate three-dimensional virtual worlds with objects and movements. Descriptions of dramatic scenes occur frequently in fiction books, describing scene contents and layout, and the movements and actions of the entities in the scene. These descriptions may be used to define graphical representations of the scene, provided they are accurately identified and interpreted. A representation of the fiction text is required that mirrors scene information as described in the book, but that is sufficient for the creation of explicit instructions for placing and moving object models in a three dimensional modeling, animation and rendering system.

We choose a *constraint-based* approach for the intermediate representation between original text and the resulting graphical instructions. We assume all entities in the fictional world are following some space-time path through the fictional universe. When a sentence is encountered that is relevant to a particular entity's position or motion, a *constraint* is created that characterises the path of the entity in some manner. For example, the sentence, "John and Mary walked side by side" would result in a constraint which limits the two entities' motions such that they are within a certain distance of each other over the period of time implied by the sentence.

1.3 Overview

Figure 1 presents an overview of the process described in this paper for creating and solving constraints from fiction text. The original text is annotated with information that points to three categories of data, namely entities (Section 3.1), relations between entities (Section 3.2) and the timelines (Section 3.3). This data is used to construct a set of abstract constraints that specify the trajectories of the object models in a 3D scene (Section 3.4). These constraints are transformed into a system of time-dependent, non-linear equations that constrain the actual trajectories of the models (Section 4). Values for the variables of each trajectory are then found using an interval-based constraint solving technique (Section 5). Thereafter the models are placed into a 3D scene, and each is animated according to its trajectory.

2. RELATED WORK

Text-to-Scene conversion research uses various ways to represent the semantic content of the story. Commonly found is the concept of a *frame* [14]: a template associated with some semantic concept. The template is hard-coded with instructions concerning its use, and contains *slots* that parameterise the instructions. The instructions indicate the way in which the frame is interpreted in producing output, and slots are filled with relevant information from the text using information extraction techniques. WordsEye [6], CarSim [21], the Put System [5], CONFUCIUS [13], and SWAN [11] are all Text-to-Scene conversion systems that make use of approaches which relate to this idea. The temporal element of text is most often described using *verbs* [11, 17, 12]. Systems based on this approach generalise all verbs into a fixed and finite set of actions [20], each of which has a frame-like representation containing slots and instructions for interpreting the action.

Temporal aspects of text-to-scene conversion are handled in different degrees by the existing systems. Conversion of language to static graphics, that is where the final result is an image rather than an animated sequence, has been explored in systems such as WordsEye [6] and the PUT system [5]. The element of time has been investigated in systems such as CarSim [1] that create animations of car accidents based on chronological event descriptions.

Solving for trajectories can be accomplished using constraint-based approaches. WordsEye [6] makes use of a constraint solver for positioning objects, but does not include a temporal aspect. CarSim [10] also makes use of constraint solving techniques for calculating trajectories of the involved vehicles. However, a sampling-based approach is used to verify trajectories, which is only suitable for its limited domain, but presents problems for conversion in more general domains such as is the case with fiction text.

None of the Text-to-Scene conversion systems documented in existing literature describe systems that are capable of working with the generalised case of fiction text, where the domain cannot be limited to a finite set of objects and actions as in the CarSim system [10]. In addition, we wish to work with original source text, as opposed to systems such as SWAN [11] and CONFUCIUS [13] which make use of constrained forms of language (namely, restricted Chinese, and dramatic scripts respectively). This paper presents a framework for handling more general cases as found in fiction text.

3. ABSTRACT CONSTRAINT FORMULATION

Abstract constraints are high level constructs that specify spatial relations between entities over specific periods of time. To create these constraints we require the identification of entities that are described in the text, as well as the spatial relationships that exist between them. In addition, the manner in which these relations occur over time must be specified in order to indicate the period over which a constraint applies.

3.1 Entities

Fiction books describe people, objects and actions that occur in a fiction or *virtual* world. We define three categories of entity that form part of this virtual world and which are capable of being represented graphically, namely *Area*, *Object* and *Avatar*. Avatars refer to characters in the story, while Objects refer to inanimate items. Areas are the spaces that Objects and Avatars may traverse.

The entities are associated with tokens or phrases in the original text as depicted in Figure 2[1]. The identification and creation of entities from fiction text is beyond the scope of this paper, but automatic approaches such as named entity extraction [22], or our earlier work on rule generalisation [8] can be used for this purpose.

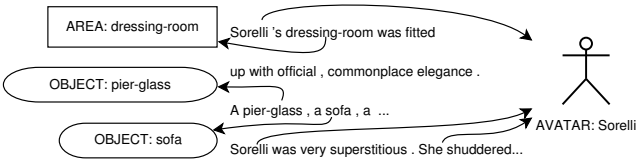


Figure 2: Illustration of the relationship between text descriptions and entities.

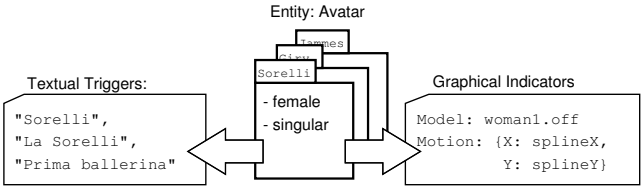


Figure 4: Illustration of the relationship between text descriptions and entities.

Each entity is associated with a list of the *textual triggers*, which are sequences of tokens in the fiction text which refer to the specific entity. In addition, each entity has fields for properties which are useful for graphical output, such as the name of an object model file attributed to the entity, and data concerning the motion of the item. Figure 4 presents an example of the *Avatar* entity. Each entity category may have attributes which characterise each instance, for example, *gender* and *multiplicity* in the Avatar case. World knowledge is encoded into the definition of these categories. In this case, Avatars are known to be humanoid

[1]Note that pronouns also act as textual triggers, but the antecedent of each pronoun is used to identify the entity that the token applies to.

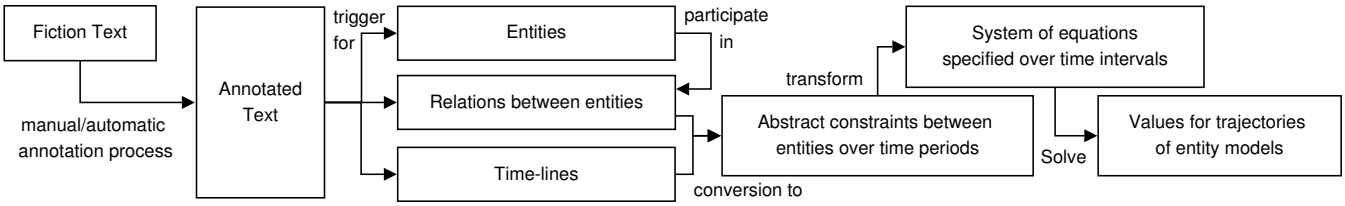


Figure 1: Overview of constraint formulation process from fiction text.

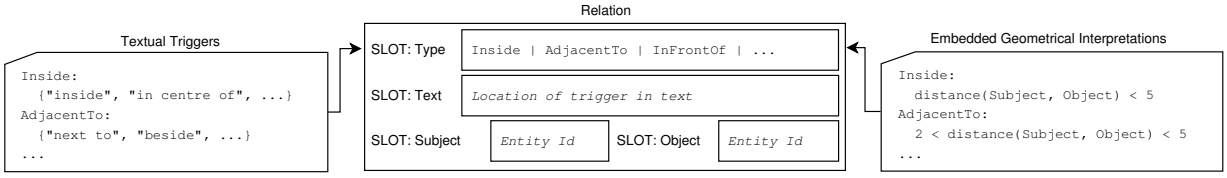


Figure 3: Indication of the slots required for the definition of a relation. Examples of the relationship between relation types textual triggers are shown, as well examples of embedded geometrical interpretations for each type of relation.

in either male or female form, which will later assist in selecting an appropriate object model.

3.2 Relations

In addition to entities, we also extract the spatial relationships between entities that are described in the text including, for example, `InFrontOf`, `AdjacentTo` and `Inside`. The structure of a Relation (shown in Figure 3) is similar to Minsky's frame [14] in that it contains instructions (graphical interpretations) and slots that parameterise these instructions. Trigger words in the text result in the creation of a relation object, and the location of the trigger in the text is recorded for use when extracting timing information. The entities acting as subject and object of the relation are identified, where the subject refers to the entity directly affected by the relation, and the object acts as a reference. Each category of relation has corresponding instructions defining its geometrical interpretation phrased as a set of constraints.

3.3 Time-lines

Fiction texts can be viewed as time based accounts describing sequences of events that occur in the virtual world. This world has its own time-line of events, and the fiction text is merely reporting on certain events that occur along this time-line. However, time is rarely quantified in fiction writing, and in most cases, an indication of when actions occur and their duration are implied. In addition, events are seldom reported in sequential order, with "flash-backs" being frequently used as a stylistic technique. Phrasings such as "he *had* been to the ..." and *"Previously,* he had done..." indicate that the action took place in the past, but with no quantification.

The non-sequential reporting of events is chosen deliberately by the book's author to create effects (such as suspense), and are crucial to the way in which the story is perceived. However, in some instances, the *actual* order of events is required to determine certain scene information. For example, the sentence in Figure 5 describes an event in which the `dressing-room` is invaded by `ballet-dancers`. Up until this point, there is no indication of where the dancers were previously, but in order to display this event graphically the original location of the dancers is required. The last portion of the sentence provides us with this information in "flash-back" form, indicating that the dancers were previously on the `stage`.

```
Suddenly the dressing-room of La Sorelli, one of
the principal dancers, was invaded by half-a-dozen
young ladies of the ballet, who had come up from
the stage after "dancing" Polyeucte.
```

Figure 5: Example of non-sequential reporting, from *The Phantom of the Opera* by Gaston LeRoux.

We derive two time-lines from a story to maintain the author's original reporting style and simultaneously maintain a chronological ordering of events in the virtual world:

- **Presentation Time-line:** This maintains the ordering of events as they are described in the original narration. The time-line is made up of *segments*, each which represent a sentence in the book. When presenting a story graphically, it is useful to be able to present the original narrative simultaneously. The segments can either be displayed graphically as subtitles, or alternatively using audio renderings of the text created using text-to-speech technology. We choose the latter, since duration of the generated audio files can be used to add timing information to each segment in the time-line.

- **Scene Time-line:** This describes the actual ordering of events in the virtual world. Consider once again Figure 5. The events in this case would be set in order if the text segment, "who had come up from the stage after dancing Polyeucte" were placed at the beginning of the sentence (assuming, that the pronoun "who" is already resolved), and some indication is made of the actual time between when the dancers were on the `stage`, and when they arrived at the `dressing-room`. Segments in the scene time-line contain parts of sentences,

which are identified manually, and assigned starting times and durations for action described. The time-line is ordered according to these specified times and duration.

The sentence in Figure 5 is presented in its equivalent time-line form in Figure 6. The presentation time-line is constructed by creating a segment in the time-line for each sentence in the fiction book. An audio file is created for the sentence using a text-to-speech converter, and timing information is derived from this file. The scene time-line is constructed by selecting portions of text and specifying the start-times and durations for each segment of text.

The presentation time-line is used to assemble the final graphical output in the original order according to the narrative. For example, the audio file and subtitles for the first segment in Figure 6 are laid down in a sequencer. The corresponding graphical output must be "filmed" from the appropriate point in the scene time-line, and this is found using the link indicated in Figure 6.

The first chapter of *The Phantom of the Opera* by Gaston LeRoux has been annotated with time-line information, and Figure 7 is a graphical representation of the two. Notice that segments in the scene time-line may overlap, indicating that events may occur concurrently in the virtual world. Most of the chapter covers a short sequence of events, but there are occasional descriptions of events of long duration. This explains why most segments in the presentation time-line map to a very small area of scene segments.

3.4 Abstract Constraint Formulation

Each entity maintains a list of relations that affect it, and this list is used to construct abstract constraints for the entity. Initially the scene time-line is traversed in order, segment by segment, and when triggers for relations are encountered the relation is added to the list of each entity it affects. In this manner, the list of relations for each entity is ordered according to the scene time-line.

An abstract constraint is created for each relation in an entity's list, specifying the subject, object and type of constraint from the slots defined in the corresponding relation. The start time is calculated with respect to the segment from the scene time-line in which the relation occurs. For instance, if the Start of a segment is specified as After previous then the preceding segment's start time is derived and added to the previous segment's duration, resulting in a starting time for the current segment. An offset from the start of the segment is calculated based on the location of the trigger in the segment, and this is added to the starting time of the segment, resulting in a start time for the abstract constraint.

The end time for each constraint is initially not set, under the assumption that a constraint holds until specified otherwise. If another constraint is applied to the entity, then the end time of the constraint is set to the start time of the succeeding abstract constraint. This prevents multiple relations applying to the same entity simultaneously, but if this is required then the duration of the relation can be explicitly added to the start time to produce the end time of the constraint.

The final result of this process is a list of abstract constraints, an example of which is presented in Figure 8. The first two constraints were generated from the example in Figure 5, and the last constraint is generated by the next sentence in the fiction text.

4. CONSTRAINED ENTITY TRAJECTORIES

Having identified a series of time-based constraints from the text, the next problem to solve is the creation of a three-dimensional scene where we place an appropriate object model for each entity, and ensure that the entity is at the correct location at the correct time according to the list of abstract constraints.

4.1 3D Model Selection

An automatic process for selecting object models for entities is dependent on a database of standardised and annotated geometry models. We have created such a database in which each model is guaranteed to be of unit height, and facing the positive $z-axis$. In addition, each model is annotated with a scaling field, which is used to scale the model in relation to other models. Most importantly each model is annotated with a set of keywords that describe it.

Each type of entity is encoded with information regarding the type of model appropriate to its representation. Avatars, for example, are encoded with instructions that only two types of model are acceptable, namely those annotated with "male" or "female", depending on the gender indicated for each specific instance (see Figure 4).

Locating models for Objects is more complex, since it is unreasonable to expect a model database to contain a corresponding model for every conceivable item in the universe. For example, our object library contains no corresponding object matching the unusual term "pier-glass". We use the lexical database WordNet [7] to resolve this issue. WordNet returns "mirror" as the immediate hypernym[2] of "pier-glass", a model of which our database does contain. It is in this spirit that objects are chosen, abstracting terms lexically, until a matching model is found.

Areas default to a model of a large cube for depicting rooms, but may also invoke scene generation procedures for outdoor environments. Examples currently under investigation are procedural terrain [16] and city generators [19].

4.2 Trajectory Creation

Each entity is assigned a time-based representation for its trajectory in the virtual world. Splines as a function of time are a convenient representation for this purpose, where an entity's location is a curve through space parameterised by a single variable t. We define the virtual world using Euclidean space in three dimensions, which means that each entity contains three splines indicating its location in time along the x, y and z axis.

Each abstract constraint is transformed into a series of expressions that define restrictions on the curves of the entities involved. These expressions are defined for each type of relation, as depicted in Figure 3.

To illustrate, assume the trajectory for an Entity i over time t is specified as $x_i(t) = a_i t^2 + b_i t + c_i$, where a_i, b_i and c_i are values that define the shape of the spline. The constraints are required to hold over a bounded and continuous span of time representing a portion of the scene time-line described in Section 3.4. Each continuous span of time is referred to as an *interval* of time T. A finite point of time within T is referred to as t.

An `Inside` constraint is specified by ensuring that the trajectories of both entities are sufficiently close for the model

[2]abstraction

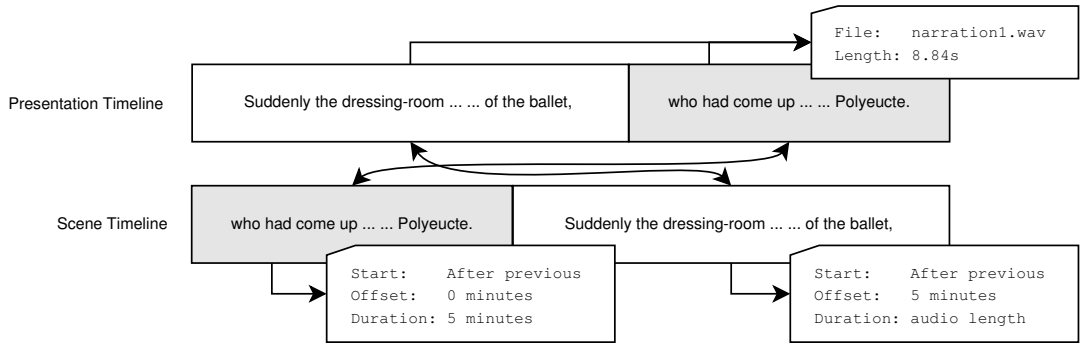


Figure 6: Relationship between presentation time-line and scene time-line.

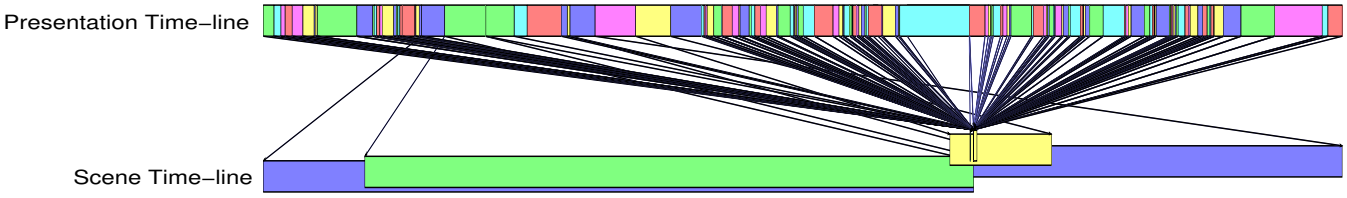


Figure 7: Presentation and scene time-line for first chapter of the *Phantom of the Opera* by Gaston LeRoux.

of the subject to be inside the object over the k-th time interval T_k:

$$(x_i(t) - x_j(t))^2 < d^2 \quad \forall t \in T_k$$

An `AdjacentTo` constraint is specified by ensuring that the entities are close, but do not intersect one another over the entire time interval T_k:

$$(x_i(t) - x_j(t))^2 > d_{min}^2 \quad \forall t \in T_k$$
$$(x_i(t) - x_j(t))^2 < d_{max}^2 \quad \forall t \in T_k$$

The bounds d_{min} and d_{max} are calculated as a function of the sizes of the bounding boxes of the affected entity object models.

Each constraint is created in the above manner, resulting in a set of expressions that describe how the curves must behave. What remains is to find suitable values of a_i, b_i and c_i for each curve that ensure that the trajectories comply with the set of constraints. Figure 9 presents a set of constraints produced from the abstract constraints in Figure 8.

5. CONSTRAINT SOLVING

We do not assume linearity for the systems of constraints that need to be solved, since we wish to specify trajectories of degree $n \geq 2$ as well as constraints that are non-linear in nature (for example involving trigonometric functions). Therefore, traditional numerical methods for solving systems of constraints, such as linear programming, are insufficient. In addition, the solving method must be capable of ensuring that each constraint is satisfied over its entire time interval.

The difficulty, as presented in Figure 9, is the presence of *universally quantified* variables, t_i. This means that a single set of parameters for the curves must be found, such that the curves satisfy the constraints for the entire specified time interval.

The time domain is a continuous one, and hence there are infinitely many real values in any given time interval. However, a computer is limited in its representation capacity, and for any given time interval, only a finite set of floating point values within the interval can be represented. The most certain method for validating a constraint would be to evaluate the constraint using a chosen set of parameters for each floating point value within the specified time interval and test that the constraint is satisfied. This would be highly inefficient, and there will still be uncertainty regarding whether constraints are valid for the non-representable real numbers. If fewer samples are evaluated then the uncertainty increases since a curve may deviate widely from the correct path between two sample points.

Interval arithmetic is a field of mathematics that works with continuous intervals, rather than on single values, making it suitable for the representation of time intervals which are specified in our set of constraints.

5.1 Interval Arithmetic

Let $\mathbb{R}$ be the set of real numbers. The set of numbers that can be represented on a computer is $\mathbb{F} \subset \mathbb{R}$. A *floating point interval* is a set of real numbers bound on either side by floating point numbers. Formally, given $g \in \mathbb{F}$ and $h \in \mathbb{F}$, then $[g, h] = \{g \leq r \leq h | r \epsilon \mathbb{R}\}$ [15]. Therefore, the interval $[g, h]$ contains every real number between (and including) g and h. An interval is denoted using uppercase (for example, $I = [g, h]$).

Let $A = [a, b]$ and $B = [c, d]$. Various interval operators are defined as *interval extensions* to their real-valued counterparts [15]:

- Addition: $A \oplus B = [a + c, b + d]$
- Subtraction: $A \ominus B = [a - d, b - c]$
- Multiplication: $A \otimes B = [min(S), max(S)]$ where $S = \{a * c, a * d, b * c, b * d\}$
- Division: $A \oslash B = A \otimes [1/d, 1/c]$ where $0 \notin B$, undefined otherwise.

CONSTRAINT:		CONSTRAINT:		CONSTRAINT:	
Subject:	ballet-dancers	Subject:	ballet-dancers	Subject:	ballet-dancers
Relation:	Inside	Relation:	Inside	Relation:	AdjacentTo
Object:	stage	Object:	dressing-room	Object:	Sorelli
Start-time:	0.0	Start-time:	0.4	Start-time:	0.4
End-time:	0.2	End-time:	0.6	End-time:	0.6

Figure 8: Example set of constraints derived from the example in Figure 5.

$$C = \left\{ \begin{array}{lll} c_1: & ((a_1t_1^2 + b_1t_1 + c_1) - (a_4t_1^2 + b_4t_1 + c_4))^2 < 12^2 & \forall t_1 \in [0.0, 0.2] \\ c_2: & ((a_1t_2^2 + b_1t_2 + c_1) - (a_3t_2^2 + b_3t_2 + c_3))^2 > 1^2 & \forall t_2 \in [0.4, 0.6] \\ c_3: & ((a_1t_2^2 + b_1t_2 + c_1) - (a_3t_2^2 + b_3t_2 + c_3))^2 < 8.9^2 & \forall t_2 \in [0.4, 0.6] \\ c_4: & ((a_1t_2^2 + b_1t_2 + c_1) - (a_2t_2^2 + b_2t_2 + c_2))^2 < 15.8^2 & \forall t_2 \in [0.4, 0.6] \end{array} \right\}$$

Figure 9: Example constraint set C, involving four objects, namely `ballet-dancers` (o_1), `dressing-room` (o_2), `Sorelli` (o_3) and `stage` (o_4) derived from the constraints specified in Figure 8. For illustrative purposes we only constrain the x dimension.

Each of the above operators can be used in place of their real-valued counterparts in order to transform an expression into its *natural interval extension*. For example, the natural interval extension of a trajectory (defined in section 4.2) is:

$$X_i(T) \equiv (A_i \otimes T \otimes T) \oplus (B_i \otimes T) \oplus C_i$$

The Cartesian Product of variables $A_i \times B_i \times C_i \times T$ is referred to as a *box* $\mathbf{B}$, and in this case forms the domain of this function.

Interval arithmetic is *inclusion monotonic*, meaning that the resulting range from evaluating the function over $\mathbf{B}$ using interval operators includes all possible values for the function over that domain. However, the reverse does not hold, and the range produced may also contain values which are not in the actual range of the function. The next section will demonstrate the difficulty that this fact introduces into constraint solving using interval arithmetic.

5.2 Interval Constraint Solving

For the constraint system C in Figure 9, we wish to find values for a_i, b_i and c_i such that all the constraints are satisfied over the specified time intervals.

5.2.1 Search Space Reduction

Initially a constraint solver is provided with a box representing the domain which has to be searched to locate solutions to the constraint system. Since the domain for each variable is an interval of real numbers, a brute force traversal of all possible combinations of values is not feasible. *Consistency techniques* [2] aid in reducing the search space by removing parts of the variable domains that invalidate the set of constraints (called *narrowing*), while still containing all the solutions. If the box does not contain a solution then a consistency technique will fail, making it a useful pruning tool.

Consistency techniques have limited narrowing abilities, and in order to tighten the bounds around a solution the narrowed box is split in half along one of its variables, and narrowing re-applied to each half. This process is continued until a box is found that validates the constraints, or until machine precision is reached (meaning that the smallest computer-representable box has been found containing a solution).

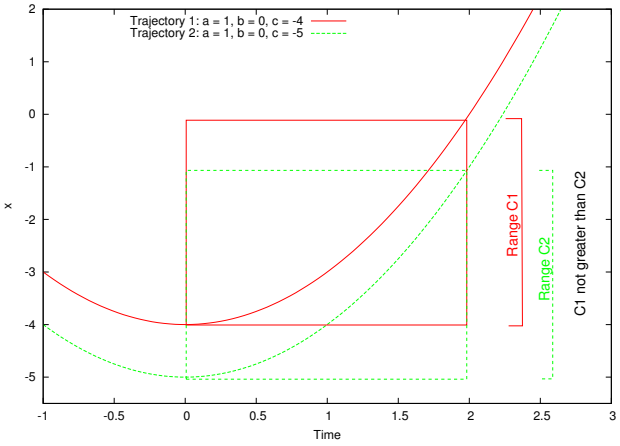


Figure 10: Illustration of the problem with evaluating a constraint over a universally quantified variable t. The constraint to be verified is $a_1t^2+b_1t+c_1 > a_2t^2 + b_2t + c_2 \; \forall t \in [0, 2]$.

5.2.2 Solutions for Universally Quantified Variables

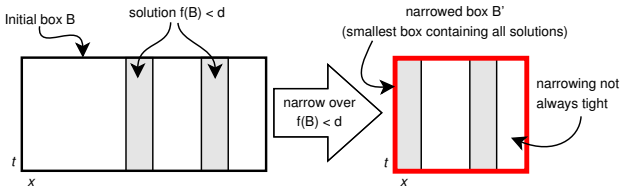
The difficulty with the universally quantified variables t_i is that we need to ensure that solutions found are valid over the entire specified interval of t_i. This means that the t_i variable may not be split. This introduces problems when checking that a box validates a constraint [9]. Consider Figure 10, which depicts a parabolic trajectory which is constrained to be above another. When the interval extension of each trajectory is evaluated over the entire time interval $[0, 2]$, the ranges indicated in the figure are returned. In order for the constraint to be satisfied the range of the top function may not overlap with the range of the bottom function. However, the figure indicates that while trajectory 1 is always above trajectory 2, the resulting ranges do not indicate this. One possible solution is to divide the interval $[0, 2]$ into smaller sections, and evaluate each section at a time, returning a positive result when all the sub-sections are satisfied. The problem with this approach is defining the minimum width of these sections, which may be required to be smaller than what is representable on a computer.

Figure 11 presents a graphical illustration of the solving process which avoids the need for evaluating a constraint to check whether a solution has been found which satisfies the universally quantified variable [3]. In Figure 11(a), the initial box $\mathbf{B}$ is narrowed over the constraint and produces box $\mathbf{B}'$ eliminating some, but not all, invalid ranges of values from the initial domains. Since narrowing operators never discard solution space, if the universally quantified variables are narrowed in any way then that means that no solutions exist in $\mathbf{B}$ which are valid over the entire interval and so this box is discarded.

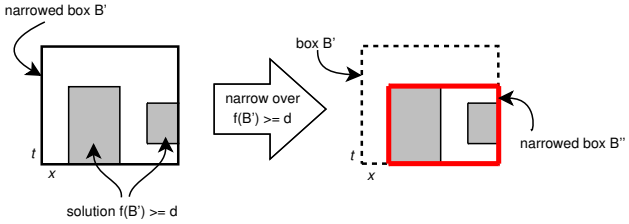
$\mathbf{B}'$ is then narrowed using the equivalent *negated* constraint, which is derived by inverting the relation operator, for example from $>$ to $\leq$. The result of this narrowing is box $\mathbf{B}''$, as illustrated in Figure 11(b).

Since the solution of the negated constraint is the inverted solution of the original constraint, any box-set difference between $\mathbf{B}'$ and $\mathbf{B}''$ is guaranteed to be a solution to the original constraint. Figure 11(c) indicates two options for calculating the box set difference. The first option yields a solution $\mathbf{P}$ that spans all t and is a universally quantified solution. If further solutions are required, then the box-set difference in option 2 is used to shrink the universally quantified interval, since the constraint is guaranteed to hold for any x over the entire sub-interval of t in $\mathbf{Q}$.

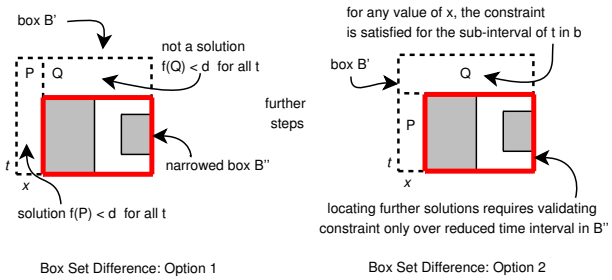
In summary, this technique is capable not only of locating solutions for universally quantified variables without requiring an evaluation step, but also enables the reduction in size of the universally quantified variable. If a solution is not encountered then box $\mathbf{B}''$ is split, and the process repeated on each sub-box.



(a) Step 1: Narrow over constraint.



(b) Step 2: Narrow over negated constraint.



(c) Step 3: Box set difference yields a solution and smaller universally quantified interval.

Figure 11: Graphical illustration of solution finding approach for constraints using universally quantified variables.

5.3 Solution and validation of Time-based Constraints

5.3.1 Implementation

We have implemented a Java version of an interval constraint solver based on the methods described in Section 5.2. The implementation allows for constraints such as those indicated in Figure 9 to be specified within text files which are parsed using Java Math Expression Parser (`http://www.singularsys.com/jep/`). We provide our own implementation of the interval operators, based on specification by Moore [15]. Expressions are evaluated using an interval implementation of function approximation using Taylor Expansion, described in Neumaier[18].

We implemented a narrowing method based on *box consistency* [4, 2] to achieve solutions of non-linear constraints containing universally quantified variables. This technique narrows domains of variables in a constraint one at a time by fixing the intervals of the remaining variables in an expression, and then finding the roots of the remaining expression (see Benhamou *et al.* [4] for a more detailed description). Although root-finding using the Interval Newton method is commonly used, we find that this method is complicated by the requirement for the evaluation of the first derivative of the function, as well as a division method which is not always successful if the denominator interval contains zero. As such we implement a method that locates roots by evaluating the interval extension of the function, only subdividing domains which evaluate to an interval containing zero, and discarding the rest. This process is repeated until machine precision is reached, at which point the smallest interval containing a root has been found. This method produces more accurate roots, and is simpler to implement, even though in some cases it requires more subdivisions than the Newton method.

We are interested in finding the first solution that satisfies the entire set of constraints, which is contrary to the algorithm described by Benhamou *et al.* [3] which is aimed at finding all solutions. Instead of finding all solutions to a single constraint, then passing these as starting search spaces to the next constraint in the system, until all constraints have been solved, we implement the algorithm in a depth-first backtracking manner, meaning that the moment a solution for a constraint is located, it is passed to the next constraint until all constraints are solved. This limits the amount of time spent on a constraint for which there are many small solutions.

The set of constraints indicated in Figure 9 is the first we have encountered in literature containing more than one universally quantified variable. The algorithm [3] is thus modified to apply to a set of constraints that contains more than one such variable. Care must be taken not to split any of these variables, and each time a box is narrowed the entire set of these variables must be checked for narrowing.

5.3.2 Configuration

The constraint set presented in Figure 9 contains 12 vari-

ables, and another two universally quantified variables. These variables are assigned the initial domains presented in Figure 12 from which the solver must find values that satisfy the constraints. Note that for object o_2 and o_4 the variable a_i and b_i are set to zero, since these are Areas and hence must have no motion. The last parameter c_i in these cases indicates the location of these Areas. The time variables $t_1$and t_2 are specially marked to indicate that they must be treated as universally quantified. The solver is set at a precision of 10^{-2}, meaning that domains may only be split if their width is greater than this value.

$$\mathbf{B} = \left\{ \begin{array}{ccc} a_1 : [7,10] & b_1 : [-1,-1] & c_1 : [-5,20] \\ a_2 : [0,0] & b_2 : [0,0] & c_2 : [29,35] \\ a_3 : [4,5] & b_3 : [-1,1] & c_3 : [-5,10] \\ a_4 : [0,0] & b_4 : [0,0] & c_4 : [-5,5] \\ t_1^* : [0,0.2] & t_2^* : [0.4,0.6] & \end{array} \right\}$$

Figure 12: Initial box for constraint system in Figure 9.

A solution is output in the form of a box containing ranges for each variable that are guaranteed to satisfy all the constraints. The mid-point of each range is substituted at the appropriate places in each expression in C (in Figure 9) and the trajectory of each object is traced from time -1 to 2. These trajectories are imported into the Blender 3D modeling package (`http://www.blender.org`), and assigned to appropriate object models for each entity (which are determined automatically as described in Section 4.1).

5.3.3 Results

The constraints in Figure 9 are solved using our constraint solver, producing trajectories indicated in Figure 13. The motion of the `ballet-dancers` and `Sorelli` are depicted as parabolae, and the graph depicts the $x - value$ of the entities' location over time. Below the graph are renderings of the actual 3D scene at certain points along the time-line.

The first rendering depicts the location of the entities before any constraints are applied. Notice that the two areas are positioned close together as an indirect result of the set of constraints (that is, the fact that the `ballet-dancers` are required to move from the `stage` to the `dressing-room` in such a short period ensures that the two Areas are positioned close together). Also at this point the `ballet-dancers` are not in the `stage` Area.

The filled rectangles in Figure 13 represent the constraints over the two time intervals. It is evident from the graph that the motion of the `ballet-dancers` satisfies constraint c_1, ensuring that they are within the `stage` for the entire period. The corresponding rendering reflects this, depicting that the `ballet-dancers` have indeed moved inside the `stage` Area.

The constraint over the second time interval is also satisfied, as is depicted with the right rectangle, indicating that the `ballet-dancers` are located inside the `dressing-room` for the entire period specified. It is clear from both the graph and the rendering however, that `Sorelli` is not in the same Area as the `ballet-dancers`, but the constraint that they be within a certain distance of each other is still satisfied. To correct this problem an additional constraint would need to be added to ensure `Sorelli` is also inside the `dressing-room` for the allocated time interval.

The final rendering depicts the scene after the constraints have lapsed, and it is evident that the path of the `ballet-dancers`, no longer constrained to be inside the `dressing-room`, leads them elsewhere.

5.3.4 Observations

The renderings in Figure 13 are an indication of the success of our constraint-based approach to Text-to-Scene conversion. It is clear that the sequence of images reflect the events described in the original sentence in Figure 5. In particular, the images show that the non-sequential manner of reporting is resolved correctly, and that the types of constraints derived are suitable for graphical representation.

The images also indicate that the mathematical expressions corresponding to each type of relation is suitable for a corresponding graphical representation, since the figures are in the correct locations at the expected times.

Finally, we observe that the interval constraint solving system successfully finds solutions to a set of constraints that contain more than one universally quantified variable.

In conclusion, Figure 13 indicates that we have created a system that is able to transform constraints derived from text into time based graphics.

5.3.5 Caveats

Currently our system finds a solution to the constraint set C in Figure 9 in an average time of 50.66 seconds on a Pentium 4 1.8GHz processor. Performance of the constraint solver depends on the complexity of the search space. This is governed by three factors:

1. The number of variables: Currently trajectories are specified as parabolae, which do not apply much flexibility in terms of interesting motion. We wish to extend these to n-degree splines, capable of flexible motions. However, such splines usually include a large number of variables, significantly extending the search space.

2. The number of dimensions: The constraints indicated in Figure 9 only apply in one dimension, suitable for representing this example. However, we aim for a 3-dimensional representation, which requires the tripling of the number of variables required to represent trajectories, and increasing the search space.

3. The number of constraints: Each sentence in a fiction book has the ability to create a number of constraints that apply to multiple trajectories over different (possibly overlapping) time intervals. The more constraints, and the more interconnected they are, the smaller the solution space becomes in a very large search space.

These scalability issues are important for our goal of representing fiction graphically, and are currently under investigation.

6. CONCLUSION

Our constraint-based method is capable of converting unrestricted fiction text into graphical output. The contributions of this paper include:

- The identification of components of fiction text which can be used as a source of objects in a graphical representation, and temporally based constraints to control the animation of these objects.

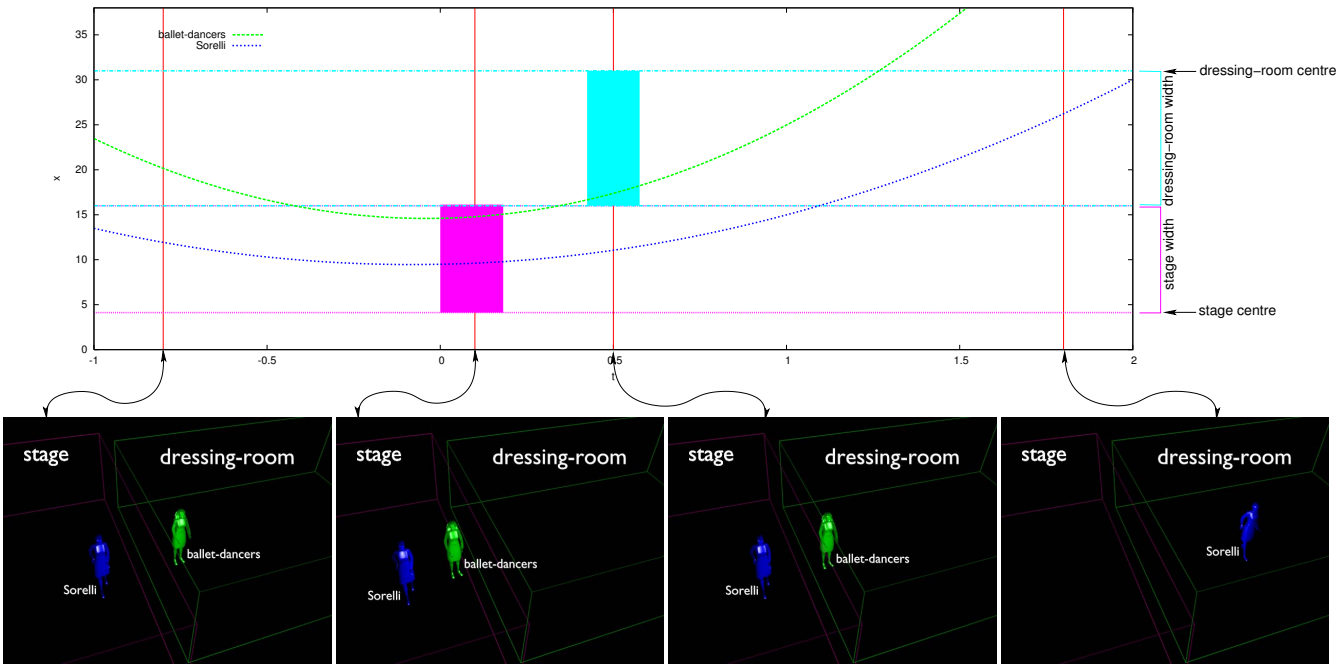


Figure 13: Interpreted solution for constraint set in Figure 9.

- A method for annotating and utilizing the presentation and scene time-line in a fiction text. The use of a text-to-speech synthesiser for deriving timings is unique.
- The constraints we describe are also unique to the Text-to-Scene domain.
- The use of interval arithmetic in solving sets of constraints has not been explored in any existing Text-to-Scene literature. We present the first example of a system containing more than one universally quantified variable, and we show that systems of these equations can be solved, guaranteeing that the trajectories satisfy the constraints over the specified time intervals.

Future work includes optimising the interval-based constraint solver in fashion that improves its scalability to handle the quantity of constraints produced by a full fiction text.

Acknowledgments

This work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom SA, Business Connexion, Comverse, Verso Technologies, Tellabs and SorTech THRIP, and the National Research Foundation. The financial assistance from the Henderson Scholarship and NRF Scarce Skills Scholarship towards this research is hereby acknowledged.

7. REFERENCES

[1] Ola Akerberg, Hans Svensson, Bastian Schulz, and Pierre Nugues. Carsim: An automatic 3D text-to-scene conversion system applied to road accident reports. In *Research Notes and Demonstrations Conference Companion, 10th Conference of the European Chapter of the Association of Computational Linguistics*, pages 191–194, Budapest, Hungary, April 12-1 2003. Association for Computational Linguistics.

[2] Frédéric Benhamou, Frédéric Goualard, Laurent Granvilliers, and Jean-Francois Puget. Revising hull and box consistency. In *ICLP'99 International Conference on Logic Programming*, pages 230–244, 1999.

[3] Frédéric Benhamou, Frédéric Goualard, 'ric Languénou, and Marc Christie. Interval constraint solving for camera control and motion planning. *ACM Transactions on Computational Logic (TOCL)*, 5(4):732–767, 2004.

[4] Frédéric Benhamou, David McAllester, and Pascal van Hentenryck. Clp(intervals) revisited. In *ILPS '94: Proceedings of the 1994 International Symposium on Logic programming*, pages 124–138, Cambridge, MA, USA, 1994. MIT Press.

[5] Sharon R. Clay and Jane Wilhelms. Put: Language-based interactive manipulation of objects. *IEEE Computer Graphics and Applications*, pages 31–39, March 1996.

[6] Bob Coyne and Richard Sproat. Wordseye: an automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496. ACM Press, 2001.

[7] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[8] Kevin Glass and Shaun Bangay. Hierarchical rule generalisation for speaker identification in fiction books. In *SAICSIT '06: Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing couuntries*,

pages 31–40, Republic of South Africa, 2006. South African Institute for Computer Scientists and Information Technologists.

[9] Frank Jardillier and Eric Languénou. Screen-space constraints for camera movements: the virtual cameraman. *Computer Graphics Forum*, 17(3):175–186, 1998.

[10] Richard Johansson, Anders Berglund, Magnus Danielsson, and Pierre Nugues. Automatic text-to-scene conversion in the traffic accident domain. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1073–1078, Edinburgh, Scotland, July 2005.

[11] Ruqian Lu and Songmao Zhang. *Automatic Generation of Computer Animation: using AI for movie animation*, volume 2160 of *Lecture Notes in Computer Science*. Springer, 2002.

[12] Minhua Ma and Paul McKevitt. Using lexical knowledge of verbs in language-to-vision applications. In *Proceedings of the 15th Artificial Intelligence and Cognitive Science Conference (AICS 2004)*, Castlebar, Ireland, September 2004.

[13] Minhua Eunice Ma. CONFUCIUS: An intelligent multimedia storytelling interpretation and presentation system. Technical report, School of Computing and Intelligent Systems, University of Ulster, Magee, September 2002.

[14] Marvin Minsky. *A framework for representing knowledge*, chapter 6, pages 211–277. The Psychology of Computer Vision. McGraw-Hill, Inc., 1975.

[15] Ramon E. Moore. *Interval Analysis*. Prentice-Hall, Inc., 1966.

[16] F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 41–50. ACM Press, 1989.

[17] A. Narayanan, D. Manuel, L. Ford, D. Tallis, and M. Yazdani. Language visualisation: Applications and theoretical foundations of a primitive-based approach. *Artificial Intelligence Review*, 9(2-3):215–235, 1995.

[18] Arnold Neumaier. *Interval Methods for Systems of Equations*, volume 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge Univ. Press, Cambridge, 1990.

[19] Yoav I. H. Parish and Pascal Muller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on computer graphics and interactive techniques*, pages 301–308. ACM Press, 2001.

[20] Roger C. Schank. The fourteen primitive actions and their inferences. Technical report, Stanford University, Stanford, CA, USA, March 1973.

[21] Hans Svensson and Ola Akerberg. Development and integration of linguistic components for an automatic text-to-scene conversion system. Master's thesis, Lund Institute of Technology, 2002.

[22] Nina Wacholder, Yael Ravin, and Misook Choi. Disambiguation of proper names in text. In *Proceedings of the fifth conference on Applied natural language processing*, pages 202–208, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.