CSCE921 Scribe Notes: January 14, 2013

Zion Schell

1 Introduction

In our previous class, we had begun to discuss temporal constraint networks. We had made the choice to examine temporal constraint networks instead of temporal logics, as temporal logics are particularly complicated (and not within the scope of this course anyway).

There are multiple methods to express temporal problems, based mainly on the dual dichotomy induced by two choices made by the planner: how time is to be represented, and how constraints on said time are to be defined. One option is to represent time as points and another is to represent time as intervals. Representing time as points makes the variables of the representative CSP be simply real numbers, representing the (very specific) time when something happens (e.g., a light turns on at time 1.3). Representing time as intervals requires the variables to be pairs of real numbers, representing the beginning and end of given events (e.g., a light begins turning on at time 1.25 and ends turning on at time 1.3).

Furthermore, the constraints can be defined either qualitatively or quantitatively. Qualitative constraints are those which only describe constraints in simple relations between two variables: "x happened before y" or "y ended at the same time as x." Quantitative constraints, on the other hand, allow the introduction of more specific constraints: "x happened five minutes before y" or "y ended within thirty seconds of x ending."

These notes will cover our discussion of qualitatively-constrained time.

2 Point Algebra (Intro)

If we use qualitative constraints and point-based representation of time, we arrive at the temporal constraint network representation called "point algebra". Point algebra uses the three basic relations "greater than" ($>$), "less than" ($<$) and "equal to" ($=$). These can be combined disjunctively for a total of $7 = 2^3 - 1$ non-null relations (for instance, the disjunctive constraint X $\{>, =\}$ Y is meant to represent "either X is greater than Y or X is equal to Y").

3 Interval Algebra (Intro)

On the other hand, if we use qualitative constraints and an interval-based representation of time, we will be using the temporal constraint network representation known as "interval algebra". Interval algebra uses a total of 13 relations which describe the relations between the beginnings and ends of two separate events. There are six of these which are invertible (and their inverses add up to another six), and one which is not invertible. They are X {"is before" (b), "overlaps with" (o), "meets" (m), "is during" (d), "starts with" (s), "finishes with" (f), "is equivalent to" ($=$)} Y and the inverses of b, o, m, d, s, and f (denoted by the addition of an i after each). These too can be disjunctively combined to arrive at $8191 = 2^{13} - 1$ non-null relations (for instance, the constraint X b, bi Y means that X is before or after Y).

4 Working with Temporal Constraint Networks

Once a problem has been correctly formulated, presumably, you would like a solution. The suggested method is to apply path consistency to the dual graph of the problem to simplify the constraints (preferably down to minimality) making a search significantly more rapid.

4.1 Path Consistency and Minimality

When using point algebra, if the constraints are convex (not containing $\neq$) the application of a single pass of the algorithm QPC-1 should reduce the dual graph to minimality. Otherwise (if the constraints contain $\neq$), achieving minimality requires 4-consistency, and it is up to the user to judge whether it would be more optimal to perform this expensive reduction or just to continue with a normal search regardless.

On the other hand, interval algebra does not react as well to path consistency algorithms. Performing said algorithm requires a 13×13 table of compositions, none of which consistently reduce the size of the problem. Of particular interest are $b \otimes bi$, $bi \otimes b$, and $d \otimes di$, three compositions which tell you precisely nothing about the relationship between the two intervals in question[1].

Once you have completed your Path-Consistency simplification of the problem, it is possible to proceed to generating a solution.

4.2 Solving a Temporal Constraint Network

If you are using point algebra, there exists a polynomial algorithm to find a solution to your problem. The algorithm uses $O(n^3)$ time and $O(n^2)$ space.

Solving interval algebra problems is in NP, that is, you must either use a general search, or use an imperfect algorithm like Allen's algorithm. Specifically, Allen's algorithm is sound but not complete[2].

5 Point Algebra v. Interval Algebra

This leaves the obvious question, why you would ever use interval algebra if it requires such a leap in complexity? The answer to this is that point algebra and interval algebra are not equivalent in expressive power. The seminal example is that in point algebra, it is not possible to express the idea "Event X, starting at time x_a and ending at time x_b is either strictly before or strictly after event Y, starting at time y_a and ending at time y_b." There is no way to preclude an overlap between time point x_b and time point y_a which does not require that X be strictly before Y using just the three relations $<$, $>$, and $=$.

However, certain elements of interval algebra problems can be reduced to equivalent point algebra problems. These are called tractible fragments. If such a fragment exists in a problem otherwise requiring interval algebra, it may be suggested to export the fragment to point algebra, solve it there, and use the results to simplify the rest of the problem's expression in interval algebra.

[1] e.g., if x is before y and y is after z, you can't tell anything about the relationship between x and z.
[2] All solutions found by the algorithm will be valid, however it is not guaranteed to find all or any solutions.

6 That Big Honkin' Composition Table

$\otimes$	$=$	b	b_i	d	d_i	f	f_i	m	m_i	o	o_i	s	s_i
$=$	$=$	b	b_i	d	d_i	f	f_i	m	m_i	o	o_i	s	s_i
b	b	b	ALL	$bmosd$	b	$bmosd$	b	b	$bmosd$	b	$bmosd$	b	b
b_i	b_i	ALL	b_i	$dfo_im_ib_i$	b_i	b_i	b_i	$dfo_im_ib_i$	b_i	$dfo_im_ib_i$	b_i	$dfo_im_ib_i$	b_i
d	d	b	b_i	d	ALL	d	$bmosd$	b	b_i	$bmosd$	$dfo_im_ib_i$	d	$dfo_im_ib_i$
d_i	d_i	$bmof_id_i$	$d_is_io_im_ib_i$	$of_id_is = s_idfo_i$	d_i	$d_is_io_i$	d_i	of_id_i	$d_is_io_i$	of_id_i	$d_is_io_i$	of_id_i	d_i
f	f	b	b_i	d	$d_is_io_im_ib_i$	f	$f_i = f$	m	b_i	osd	$o_im_ib_i$	d	$o_im_ib_i$
f_i	f_i	b	$d_is_io_im_ib_i$	osd	d_i	$f_i = f$	f_i	m	$d_is_io_i$	o	$d_is_io_i$	o	d_i
m	m	b	$d_is_io_im_ib_i$	osd	b	osd	b	b	$f_i = f$	b	osd	m	m
m_i	m_i	$bmof_id_i$	b_i	dfo_i	b_i	m_i	m_i	$s = s_i$	b_i	dfo_i	b_i	dfo_i	b_i
o	o	b	$d_is_io_im_ib_i$	osd	$bmof_id_i$	osd	bmo	b	$d_is_io_i$	bmo	$of_id_is = s_idfo_i$	o	of_id_i
o_i	o_i	$bmof_id_i$	b_i	dfo_i	$d_is_io_im_ib_i$	o_i	$d_is_io_i$	of_id_i	b_i	$of_id_is = s_idfo_i$	$o_im_ib_i$	dfo_i	$o_im_ib_i$
s	s	b	b_i	d	$bmof_id_i$	d	bmo	b	m_i	bmo	dfo_i	s	$s = s_i$
s_i	s_i	$bmof_id_i$	b_i	dfo_i	d_i	o_i	d_i	of_id_i	m_i	of_id_i	o_i	$s = s_i$	s_i