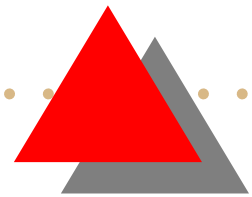
A Database Transaction Scheduling Tool in Prolog

Steve Barker & Paul Douglas



Overview of Talk

- Motivations;
- CRAS Properties;
- Our Courseware (Key Features and Example);
- Test results;
- Future Work.

Some Motivations

To develop an item of courseware to:

- Encourage student-centered learning;
- Learning by hypothesis formulation;
- Provide intelligent diagnosis of errors and explanation facilities;
- . . .

Aid students learning about the CRAS properties.

CRAS Properties: CS

A schedule σ on a set of transactions T is CS iff:

$$\forall t_i, t_j \in T(\sigma)\ conflict(t_i, t_j) \rightarrow \neg conflict(t_j, t_i)$$

where conflict is defined thus:

$$\forall t_i, t_j \in T(\sigma)\ r_i(x) < w_j(x) \rightarrow\ conflict(t_i, t_j)$$

$$\forall t_i, t_j \in T(\sigma)\ w_j(x) < r_i(x) \rightarrow\ conflict(t_j, t_i)$$

$$\forall t_i, t_j \in T(\sigma)\ w_i(x) < w_j(x) \rightarrow\ conflict(t_i, t_j).$$

CRAS Properties: RC & ACA

A schedule σ on a set of transactions T is recoverable iff:

$$\forall t_i, t_j \in T(\sigma)\ read\_from(t_i, t_j) \rightarrow$$
$$c_j \in \sigma\ \wedge\ c_j < c_i$$

A schedule σ on a set of transactions T avoids cascading aborts iff:

$$\forall t_i, t_j \in T(\sigma)\ read\_from(t_i, t_j) \rightarrow$$
$$c_j < r_i(x) \vee a_j < r_i(x)$$

CRAS Properties: ST

A schedule σ on a set of transactions T is strict iff:

$$\forall t_i, t_j \in T(\sigma)\ w_j(x) < r_i(x) \vee w_j(x) < w_i(x) \rightarrow$$
$$a_j < r_i(x) \vee c_j < r_i(x) \vee$$
$$a_j < w_i(x) \vee c_j < w_i(x)$$

Read Froms

The auxiliary predicate read_from is defined thus:

$$\forall t_i, t_j \in T(\sigma)\ \exists x[read\_from(t_i, t_j) \leftarrow w_j(x) < r_i(x) \wedge \neg(a_j < r_i(x)) \wedge [\forall t_k \in T(\sigma)\ w_j(x) < w_k(x) < r_i(x) \rightarrow a_k < r_i(x)]].$$

Our Software: Key Features

Design based on:

- Gagne's event-based theory of instruction;
- Phenomenographic studies of student learners.

Implementation in Prolog (XSB and Sicstus).

Implementation now includes a Web-based front-end.

Implementation

Each rule for CRAS is translated into Prolog:

$$conflict(N, M) :- o(r, N, Y, T1),$$
$$o(w, M, Y, T2), T1 < T2, N = \backslash = M.$$

$$conflict(M, N) :- o(r, N, Y, T1),$$
$$o(w, M, Y, T2), T2 < T1, N = \backslash = M.$$

$$conflict(N, M) :- o(w, N, Y, T1),$$
$$o(w, M, Y, T2), T1 < T2, N = \backslash = M.$$

Schedule Representation

A schedule is a sequence:

$$\langle r_1(x), w_2(y), \ldots, \rangle.$$

Represented by a set of facts (with timestamp):

$$\{o(r, 1, x, t_1), o(w, 2, y, t_2), \ldots, \} \quad (where\ t_1 < t_2).$$

Example

w,1,x,90

w,1,y,95

r,2,u,100

w,1,z,105

w,2,z,110

c,1,null,115

w,2,x,120

r,2,y,125

w,2,y,130

c,2,null,135

?-aca.

Test Results (Summary)

Formative and summative evaluations undertaken.

Summative evaluation (Likert scale and t-test) reveals:

- Statistically significant results for **perceived** value (relative to standard text).

Future Work

- Extension of existing software e.g., other properties.
- Longitudinal evaluation of current system.
- Other applications of the software in Computer Science curriculum.