SPECIFYING AND VERIFYING CORRECTNESS OF TRIGGERS USING DEclarATIVE LOGIC

Chitta Baral

Department of Computer Science & Engineering
Arizona State University
Tempe, Arizona, USA

Chitta Baral (joint work with Mutsumi Nakamura)

October 15, 2001

http://www.public.asu.edu/~cbaral/
### Triggers: Event-Condition-Action (ECA) Rules

<table>
<thead>
<tr>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>Service</td>
</tr>
<tr>
<td>27</td>
<td>Accounting</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>...</th>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounting</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>Services</td>
<td>Peter</td>
<td></td>
</tr>
<tr>
<td>Administration</td>
<td>Mary</td>
<td></td>
</tr>
<tr>
<td>Accounting</td>
<td>John</td>
<td></td>
</tr>
</tbody>
</table>

#### EMP

- John: DeptName, EmpName, EmpId
- Doug: DeptName, EmpName, EmpId
- Mary: DeptName, EmpName, EmpId

#### DEPT

- Accounting: DeptName, ManagerId
- Service: DeptName, ManagerId

#### Relational Database: a bunch of tables (relational instances)

### Triggers and active databases
Certain update to the database trigger additional updates as dictated by the ECA rules.

- An ECA rule
- Event: Deletion of a tuple in the EMP table
- Condition: The EmpId in that tuple appears as a ManagerId in the DEPT table
- Action: Remove all such tuples in the DEPT table.
Introduction

Current status: DB systems that have triggers and their usage

Available in most recent database systems: IBM DB2/V2, Oracle, etc.

But rarely used.

Need to be able to specify the purpose.

Need to be able to verify the correctness.

Correct! Correct with respect to what?

But dangerous: automatically changes other tables.

What does it mean that a set of triggers is not dangerous?

How do we state the purpose? In what language?

What is the purpose of a set of triggers?

Available in most recent database systems: IBM DB2/V2, Oracle, etc.

Specifying and Verifying Correctness of Triggers

Introduction
It would be great if certain triggers could be automatically generated from the specification.
Specifying and verifying correctness of triggers

Evolution of a database due to updates and triggers

Semantics, Specification, and Correctness

Updates and actions: Insert a tuple, delete a tuple, modify or update a tuple.

Semantics: A function from states and action sequences to a sequence of states.

Notations:
- \( \sigma \): denotes the last state of the evolution given by \( \Phi \).
- \( \sigma(\alpha_1, \alpha_2) \): denotes the last state of the evolution given by \( \Phi(\sigma, \alpha_1, \alpha_2) \).

We similarly define operations on the set of triggers.

We denote \( \Phi \) by \( \Phi^L \) for the evolution of the database given a sequence of actions and triggers.

Updates and actions: Insert a tuple, delete a tuple, modify or update a tuple.

Evolution of a database due to updates and triggers
Specifying and verifying correctness of triggers

Semantics, Specification, and Correctness

Specifying ideas

- Four kinds: state invariance constraints; state maintenance constraints; trajectory invariance constraints; and trajectory maintenance constraints.

A state constraint \( \gamma_s \) on a database scheme \( H \) is a function that associates with each database \( r \) of \( H \) a boolean value \( \gamma_s(r) \). A database \( r \) of \( H \) is said to satisfy \( \gamma_s \) if \( \gamma_s(r) \) is true.

A trajectory constraint \( \gamma_t \) on a database scheme \( H \) is a function that associates with each database sequence \( \Upsilon \) of \( H \) a boolean value \( \gamma_t(\Upsilon) \). A database sequence \( \Upsilon \) of \( H \) is said to satisfy \( \gamma_t \) if \( \gamma_t(\Upsilon) \) is true.

Specifying ideas
Specifying and verifying correctness of triggers

Semantics, Specification, and Correctness

The specification in our example: "For any tuple t in the DEPT table, there must be a tuple t' in the EMP table such that t.ManagerId = t'.EmpId" is true in all quiescent states. (a trajectory maintenance constraint)
Let $\Gamma_i$ be a set of state invariant constraints, $\Gamma_m$ be a set of state maintenance constraints, $\Gamma_t$ be a set of trajectory invariant constraints, and $\Gamma_{tm}$ be a set of trajectory maintenance constraints. Let $A$ be a set of exogenous actions.

We say $L$ is correct with respect to $\Gamma_i \cup \Gamma_m \cup \Gamma_t \cup \Gamma_{tm}$ and $A$ if for all database states where $\sigma$ holds, and action sequences $\alpha_1, \ldots, \alpha_n$ consisting of exogenous actions from $A$, all the states $\sigma_{\alpha_1}, \ldots, \sigma_{(\alpha_1, \ldots, \alpha_{n-1})\alpha_n}$ satisfy the constraints in $\Gamma_i$; all the states satisfy the constraints in $\Gamma_m$ with $\{\alpha_i\}$; the trajectory obtained by concatenating $\psi(\sigma, \alpha_1)$ with $\psi(\sigma_{\alpha_1}, \alpha_2)$, $\ldots$, $\psi(\sigma_{(\alpha_1, \ldots, \alpha_{n-1})\alpha_n}, \alpha_n)$ satisfies the constraints in $\Gamma_t$; and the trajectory $\sigma, \sigma_{\alpha_1}, \ldots, \sigma_{(\alpha_1, \ldots, \alpha_{n-1})\alpha_n}$ satisfies the constraints in $\Gamma_{tm}$. 

Definition of Correctness

Semantics, Specification, and Correctness
USING DLP FOR SPECIFICATION, SIMULATION AND VERIFICATION
Specifying and verifying correctness of triggers using DLP for specification, simulation and verification

Declarative Logic Programming (DLP)

• Semantics: Given in terms of answer sets.
  \( a_0 \leftarrow a_1, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n \) can be assumed to be false when \( a_0 \) must be true.

• Intuitive meaning: If \( a_1, \ldots, a_m \) are true and \( a_{m+1}, \ldots, a_n \) can be assumed \( \text{false} \), then \( a_0 \) must be true.

where \( a_i \)'s are atoms.

A DLP is a collection of rules of the form

\( a_0 \rightarrow a_1, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n \)
Specifying and verifying correctness of triggers.

Using DLP for specification, simulation, and verification.

An illustration: the database schema and the specification

- The schema

  purchase (purchase id, client, amount).
  payment (payment id, client, amount).
  account (client, credit, status).

Each client and their credit status.

The relation account stores the available credit for each client. The relation purchase and payment records the purchase history of clients. The relation payment is a foreign key with respect to the relation account. The relation purchase records the purchase history of clients.

The underlined attributes are the primary keys and the attribute client account (client, credit, status).

The schema

Specifying and verifying correctness of triggers.
Specifying and verifying correctness of triggers using DLP for specification, simulation, and verification

State maintenance constraints:

1. For each client $c$ which appears in a tuple $a$ in the relation account: if $a$.credit $\geq 3K$ then $a$.status = good, and if $a$.credit $< 3K$ then $a$.status = bad.

2. For each client $c$ which appears in a tuple $a$ in the relation account: $a$.credit is $5K$ minus the sum of all the purchase amounts for $c.$
Specifying and verifying correctness of triggers

Using DLP for specification, simulation, and verification

**Triggers:**

- **Trigger 1:** When a tuple \( p \) is added to the relation `purchase` such that \( p\text{.client} = a\text{.client} \), then \( a\text{.credit} \) is updated so that the updated \( a\text{.credit} \) has the value obtained by subtracting \( p\text{.amount} \) from the old \( a\text{.credit} \).

- **Trigger 4:** When a tuple \( a \) in the relation `account` is updated such that \( a\text{.credit} \) is more than or equal to 3K, then \( a\) is further updated such that \( a\text{.status} \) has the value "good".

- **Trigger 2:** When a tuple \( p \) in the relation `payment` is added to the `payment` relation, then the tuple \( a \) in the relation `account` such that \( p\text{.client} = a\text{.client} \) is updated so that the updated \( a\text{.credit} \) has the value obtained by subtracting \( p\text{.amount} \) from the old \( a\text{.credit} \).

- **Trigger 3:** When a tuple \( a \) in the relation `account` is updated such that \( a\text{.credit} \) is less than 3K, then \( a\text{.status} \) has the value "bad".

- **Trigger 5:** When a tuple \( a \) in the relation `account` is updated such that \( a\text{.credit} \) is more than or equal to 3K, then \( a\) is further updated such that \( a\text{.status} \) has the value "good".
Specifying and verifying correctness of triggers

Using DLP for specification, simulation and verification

A general methodology and an illustration

Step 1: Representing the initial state (Π₀)

\[
\begin{align*}
  &\text{holds}(\text{account}(c', 2', \text{good}), 1), \\
  &\text{holds}(\text{account}(q', 1', \text{bad}), 1), \\
  &\text{holds}(\text{account}(a', 3', \text{good}), 1), \\
  &\text{holds}(\text{payment}(2', q, 1), 1), \\
  &\text{holds}(\text{payment}(1', a, 1), 1), \\
  &\text{holds}(\text{payment}(1', b, 1), 1), \\
  &\text{holds}(\text{purchase}(2', q', 1'), 1), \\
  &\text{holds}(\text{purchase}(1', a', 3'), 1), \\
  &\text{holds}(\text{purchase}(1', c, 5), 1).
\end{align*}
\]
Specifying and verifying correctness of triggers
Using DLP for specification, simulation and verification

Step 1: Enumerating the possible initial states \( \Pi_{\text{enum}} \)

\[
(Z,X)^{\text{account}} \not\text{holds}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]

\[
(Z)^{\text{status}} \not\text{dom}(X)^{\text{amount}}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]

\[
(Z)^{\text{payment}} \not\text{holds}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]

\[
(Z)^{\text{payment}} \not\text{dom}(X)^{\text{amount}}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]

\[
(Z)^{\text{payment}} \not\text{holds}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]

\[
(Z)^{\text{payment}} \not\text{dom}(X)^{\text{amount}}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]

\[
(Z)^{\text{payment}} \not\text{holds}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]

\[
(Z)^{\text{purchase}} \not\text{holds}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]

\[
(Z)^{\text{purchase}} \not\text{dom}(X)^{\text{amount}}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]

\[
(Z)^{\text{purchase}} \not\text{holds}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]

\[
(Z)^{\text{purchase}} \not\text{dom}(X)^{\text{amount}}
\quad \to \quad (1,(Z,X)^{\text{account}})
\]
Specifying and verifying correctness of triggers

Using DLP for specification, simulation and verification

\( (Z \cdot \lambda \cdot X, \text{amount} \cdot (X) \cdot \text{status} \cdot (Z \cdot \lambda \cdot X) \cdot \text{dom} \cdot (X) \cdot \text{same} ) \)

\( \rightarrow ((Z \cdot \lambda \cdot X) \cdot \text{account} \cdot (Z \cdot \lambda \cdot X) \cdot \text{dom} \cdot (X) \cdot \text{same} ) \)

\( (Z) \cdot \text{ payments} \cdot (Z \cdot \lambda \cdot X) \cdot \text{dom} \cdot (X) \cdot \text{id} \cdot \text{ same} ) \)

\( \rightarrow ((Z \cdot \lambda \cdot X) \cdot \text{ payments} \cdot (Z \cdot \lambda \cdot X) \cdot \text{dom} \cdot (X) \cdot \text{id} \cdot \text{ same} ) \)

\( (Z) \cdot \text{ purchases} \cdot (Z \cdot \lambda \cdot X) \cdot \text{dom} \cdot (X) \cdot \text{id} \cdot \text{ same} ) \)

\( \rightarrow ((Z \cdot \lambda \cdot X) \cdot \text{ purchases} \cdot (Z \cdot \lambda \cdot X) \cdot \text{dom} \cdot (X) \cdot \text{id} \cdot \text{ same} ) \)

\( (\lambda \cdot X) \cdot \text{ not initially poss } \rightarrow (\lambda \cdot X) \cdot \text{ not initially poss } \)

\( (\lambda \cdot X) \cdot \text{ initially possible } \rightarrow (\lambda \cdot X) \cdot \text{ initially possible } \)

\( \begin{align*}
\text{Step 2: } & \text{ Enumerating the initial exogenous actions (}\Pi_{\text{occ}}\text{)} \\
\text{occurs} \cdot \text{ins} \cdot \text{purchase}(\xi, c', \xi', 1) \end{align*} \)

\( \begin{align*}
\text{Step 2': } & \text{ Action occurrence in the initial state (}\Pi_{\text{occ}}\text{)} \\
\text{not holds} \cdot \text{account}(\lambda, \text{dom} \cdot (X) \cdot \text{status} \cdot (Z \cdot \lambda \cdot X) \cdot \text{dom} \cdot (X) \cdot \text{same} ) \)

\( \rightarrow ((Z \cdot \lambda \cdot X) \cdot \text{account} \cdot (Z \cdot \lambda \cdot X) \cdot \text{dom} \cdot (X) \cdot \text{same} ) \)
Specifying and verifying correctness of triggers

Using DLP for specification, simulation and verification

\( (\mathcal{L}, (Z \cdot \lambda \cdot X)) \text{holds} \rightarrow (\mathcal{L}, (Z \cdot \lambda \cdot X) \text{del, account}) \rightarrow (\mathcal{L}, (Z \cdot \lambda \cdot X) \text{ins, account}) \)

\( (\mathcal{L}, (Z \cdot \lambda \cdot X) \text{del, payment}) \rightarrow (\mathcal{L}, (Z \cdot \lambda \cdot X) \text{ins, payment}) \)

\( (\mathcal{L}, (M \cdot \lambda \cdot X) \text{purchase}) \rightarrow (\mathcal{L}, (M \cdot \lambda \cdot X) \text{del, purchase}) \rightarrow (\mathcal{L}, (M \cdot \lambda \cdot X) \text{ins, purchase}) \)

Step 4: Executability (\( \Pi_{\text{exec}}^{\text{e}} \))

\( (x + 1 \cdot \mathcal{L}, F, q \cdot \text{out, occurred}) \text{holds} \rightarrow (1 + \mathcal{L}, F, q) \)

\( (\mathcal{L}, \text{del, pd}, F, C) \text{executables} \rightarrow (\mathcal{L}, \text{ins, pd}, F, C) \text{executables} \)

\( (\mathcal{L}, \text{del, pd}, F, C) \text{executables} \rightarrow (\mathcal{L}, \text{ins, pd}, F, C) \text{executables} \)

\( (\mathcal{L}, \text{del, account}) \text{executables} \rightarrow (\mathcal{L}, \text{ins, account}) \text{executables} \)

Step 3: Effect of actions and inertia (\( \Pi_{\text{eff}}^{\text{i}} \))

\( (\lambda \cdot X) \text{initially} \rightarrow (\lambda \cdot Y) \text{initially} \)
specifying and verifying correctness of triggers

using dlp for specification, simulation and verification

\[ (\mathcal{L}'(S' \lambda X)) \text{ occurs, account} \]
\[ \text{good} = S' \varepsilon > \lambda \rightarrow (1 + \mathcal{L}'(\text{account})) \text{ holds, account} \]
\[ (\mathcal{L}'(S' \lambda X)) \text{ occurs, account} \]
\[ \text{good} = S' \varepsilon > \lambda \rightarrow (1 + \mathcal{L}'(S' \lambda X)) \text{ holds, account} \]
\[ (\mathcal{L}'(\lambda W'X)) \text{ occurs, purchase} \]
\[ \rightarrow (1 + \mathcal{L}'(S'B'X)) \text{ holds, account} \]
\[ (\mathcal{L}'(\lambda W'X)) \text{ occurs, purchase} \]
\[ \rightarrow (1 + \mathcal{L}'(S'B'X)) \text{ holds, account} \]

\[ (\Pi) \text{ steps trigers} \]

\[ \rightarrow (\mathcal{L}'(Z' \lambda X)) \text{ occurs, account} \]
\[ \rightarrow (\mathcal{L}'(Z' \lambda X)) \text{ occurs, account} \]

\[ \text{executable (upd, account)} \]

\[ \rightarrow (\mathcal{L}'(Z' \lambda X)) \text{ occurs, account} \]
\[ \rightarrow (\mathcal{L}'(Z' \lambda X)) \text{ occurs, account} \]

\[ \text{executable (upd, account)} \]
Specifying and verifying correctness of triggers

Using DLP for specification, simulation, and verification

Step 6: Identifying quiescent states (Φq)

\[ Φq = S, τ \leq λ \rightarrow (1 + Φ(S, λ, X)) \]

\[ Φq = S, τ < λ \rightarrow (1 + Φ(S, λ, X)) \]

\[ Φq = S, τ \leq λ \rightarrow (1 + Φ(S, λ, X)) \]

\[ Φq = S, τ < λ \rightarrow (1 + Φ(S, λ, X)) \]

\[ Φq = S, τ \leq λ \rightarrow (1 + Φ(S, λ, X)) \]

\[ Φq = S, τ < λ \rightarrow (1 + Φ(S, λ, X)) \]

\[ Φq = S, τ \leq λ \rightarrow (1 + Φ(S, λ, X)) \]

\[ Φq = S, τ < λ \rightarrow (1 + Φ(S, λ, X)) \]
Specifying and verifying correctness of triggers

Using DLP for specification, simulation, and verification

Step 7: Defining domains ($\Pi_{dom}$)

- $\text{fluent}(\text{purchase}(X,Y,W)) \leftarrow \text{id}_{dom}(X), \text{cname}_{dom}(Y), \text{amount}(W)$.

- $\text{fluent}(\text{payment}(X,Y,Z)) \leftarrow \text{id}_{dom}(X), \text{cname}_{dom}(Y), \text{amount}(Z)$.

- $\text{fluent}(\text{account}(X,Y,Z)) \leftarrow \text{cname}_{dom}(X), \text{amount}(Y), \text{status}(Z)$.

Step 8: Specification ($\Pi_{cons}$)

- $(X)\text{amount}_{sum}\leftarrow (\lambda)\text{time}_{sum}(C, \Sigma, T) \rightarrow \text{time}_{(\lambda)\Sigma}(C, \Sigma, T)$

- $(X)\text{amount}_{sum}\leftarrow (\lambda)\text{id}_{dom}(X), \text{purchase}_{total}(C, \Sigma, T)$

- $(X)\text{amount}_{sum}\leftarrow (\lambda)\text{time}_{sum}(C, \Sigma, T)$

- $(Z)\text{amount}_{sum}(X)\leftarrow (X)\text{amount}_{sum}(X), \text{status}_{account}(Z, X, \lambda)$

- $(Z)\text{amount}_{sum}(X)\leftarrow (Z, X, \lambda)\text{amount}_{account}(Z, X, \lambda)$

- $(Z)\text{amount}_{sum}(X)\leftarrow (Z, X, \lambda)\text{fluent}_{amount}(Z, X, \lambda)$

- $(Z)\text{amount}_{sum}(X)\leftarrow (Z, X, \lambda)\text{fluent}_{amount}(Z, X, \lambda)$

- $(Z)\text{amount}_{sum}(X)\leftarrow (Z, X, \lambda)\text{fluent}_{purchase}(Z, X, \lambda)$

- $(Z)\text{amount}_{sum}(X)\leftarrow (Z, X, \lambda)\text{fluent}_{purchase}(Z, X, \lambda)$

- $(Z)\text{amount}_{sum}(X)\leftarrow (Z, X, \lambda)\text{fluent}_{purchase}(Z, X, \lambda)$

- $(Z)\text{amount}_{sum}(X)\leftarrow (Z, X, \lambda)\text{fluent}_{purchase}(Z, X, \lambda)$

- $(Z)\text{amount}_{sum}(X)\leftarrow (Z, X, \lambda)\text{fluent}_{purchase}(Z, X, \lambda)$

- $(Z)\text{amount}_{sum}(X)\leftarrow (Z, X, \lambda)\text{fluent}_{purchase}(Z, X, \lambda)$
Using DLP for specification, simulation and verification

Theorem 1: \( \Pi_{\text{correct}} \subseteq \Pi_{\text{correct}} \cup \Pi_{\text{cons}} \cup \Pi_{\text{ex}} \cup \Pi_{\text{tr}} \cup \Pi_{\text{qu}} \cup \Pi_{\text{dom}} \cup \Pi_{\text{spec}} \)

Theorem 2: \( \Pi_{\text{not correct}} \)

\[ \lambda = (L, \lambda, C, X) \]
\[ \lambda = (L, \lambda, C, X) \]

\[ \text{weight} \quad \text{holds} \]
\[ \text{weight} \quad \text{holds} \]

\[ \text{correct} \quad \text{not correct} \]

\[ \text{not correct} \quad \text{main construct} \rightarrow \text{correct} \]

\[ \text{violation}(\text{X}) \quad \text{quiescent}(\text{L}) \quad \text{violated}(\text{L}) \]

\[ \text{holds account}(C, Cr, bad) \rightarrow \text{true} \]

\[ \text{holds account}(C, Cr, good) \rightarrow \text{true} \]

\[ \text{violated}(C, L) \rightarrow \text{false} \]

\[ \text{purchase total}(C, \text{Sum2}, L) \quad \text{payment total}(C, \text{Sum1}, L) \]

\[ \text{violated}(C, L) \rightarrow \text{false} \]

Theorem 1: \( \Pi_{\text{correct}} \subseteq \Pi_{\text{correct}} \cup \Pi_{\text{cons}} \cup \Pi_{\text{ex}} \cup \Pi_{\text{tr}} \cup \Pi_{\text{qu}} \cup \Pi_{\text{dom}} \cup \Pi_{\text{spec}} \)

Theorem 2: \( \Pi_{\text{not correct}} \)
• Automatic generation of triggers. (before triggers, after triggers)
• More complicated triggers.
• Inferring events.
• Various execution models. (deferred, immediate)
• More general constraints; temporal constructs.

**Next Steps:** Some in the paper
Specifying and verifying correctness of triggers

Using DLP for specification, simulation and verification

Conclusion