Abstract

SQL is a relational database definition and manipulation language. Portions of the manipulation language are readily described in terms of relational algebra. The semantics of a subset of the SQL select statement is described. The select statement allows the user to query the database. The select statement is shown to be equivalent to a series of relational and set operations. The semantics are described in terms of abstract data types for relation schemes, tuples, and relations. Certain forms of the union or intersection of two select statements are shown to have equivalent single select statement forms.
# Semantic Definition of the SQL select Statement

## Contents

1. **Introduction**

2. **The SQL select Statement**
   - 2.1 Abstract Syntax .......................... 2
   - 2.2 Concrete Syntax .......................... 2
   - 2.3 Notation ................................. 4

3. **Semantic Definition**
   - 3.1 Semantic Techniques .................. 4
   - 3.2 Informal Semantics .................... 5
   - 3.3 The Relational Model .................. 8
   - 3.4 Formal Semantics ...................... 10
   - 3.5 Valuation Functions ................... 15

4. **Equivalence of Queries** .................. 19

5. **Conclusions** ............................. 19
1 Introduction

The Structured Query Language (SQL) consists of a set of commands and utilities for defining and manipulating a relational database. A relational database is viewed by the user as a collection of relations or tables. In fact, these two terms are often used interchangeably.

A table is an unordered collection of rows, or tuples. Each element of a row is generally referred to as an attribute or column. The degree or arity of a table is the number of columns in the table. The cardinality of a table is the number of tuples in the table.

One of the first implementations of SQL was in the early 1980's as an interface to a relational database management system called System/R [2, 7]. In 1986, the American National Standards Institute (ANSI) adopted a standard syntax and English language semantics[1].

2 The SQL select Statement

The subset of SQL to be considered here is the select statement. This subset of SQL allows the user to make queries about a previously defined database (a set of relations).

2.1 Abstract Syntax

<table>
<thead>
<tr>
<th>$\Psi \in$ Query</th>
<th>$\epsilon \in$ Expression</th>
<th>$\xi \in$ Identifier</th>
<th>$I \in$ Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma \in$ Selection</td>
<td>$\Pi \in$ Predicate</td>
<td>$S \in$ String</td>
<td>$R \in$ Real</td>
</tr>
<tr>
<td>$\Gamma \in$ Table Expression</td>
<td>$T \in$ Table List</td>
<td>$C \in$ Column</td>
<td>$B \in$ Boolean</td>
</tr>
</tbody>
</table>

- $\Psi \Rightarrow \text{select } \Theta \mid \Psi \text{ union } \Psi \mid \Psi \text{ intersection } \Psi$
- $\Theta \Rightarrow \Sigma \Gamma$
- $\Sigma \Rightarrow * \mid \epsilon \mid \Sigma$
- $\epsilon \Rightarrow \epsilon + \epsilon \mid \epsilon - \epsilon \mid \epsilon * \epsilon \mid \epsilon / \epsilon \mid + \epsilon \mid - \epsilon \mid (\epsilon) \mid R \mid I \mid S \mid C$
- $\Rightarrow \epsilon \mid \epsilon \text{ or } \epsilon \mid \epsilon \text{ and } \epsilon \mid \text{ not } \epsilon \mid \Pi \mid (\epsilon)$
- $\Gamma \Rightarrow \text{from } T \mid \text{from } T \text{ where } \epsilon$
- $T \Rightarrow \xi \mid \xi \mid \xi \mid T \mid T$
- $C \Rightarrow \xi \mid \xi \mid \xi \mid T \mid T$
- $\Pi \Rightarrow \epsilon \mid \Omega \epsilon$
- $\Omega \Rightarrow = \mid < \mid > \mid <= \mid >=$

2.2 Concrete Syntax

The concrete syntax is taken from the ANSI standard [1] and is given in Table 1 and Table 2. Any deviations from the SQL standard syntax are to remove portions of syntax unrelated to the select statement and to remove portions of select statement syntax that will not be implemented.
Semantic Definition of the SQL select Statement

Table 1: Common Elements Used in the Query Statement Syntax (adapted from the ANSI SQL Standard)

\[
\begin{array}{l}
\text{<uppercase>} 
\Rightarrow A \mid B \mid C \mid \cdots \mid X \mid Y \mid Z \\
\text{<lowercase>} 
\Rightarrow a \mid b \mid c \mid \cdots \mid x \mid y \mid z \\
\text{<digit>} 
\Rightarrow 1 \mid 2 \mid 3 \mid \cdots \mid 8 \mid 9 \mid 0 \\
\text{<letter>} 
\Rightarrow \text{<uppercase>} \mid \text{<lowercase>} \\
\text{<character>} 
\Rightarrow \text{<letter>} \mid \text{<digit>} \\
\text{<ident>} 
\Rightarrow \text{<letter>} \mid \{ \mid \text{<letter>} \mid \text{<digit>} \} \star \\
\text{<literal>} 
\Rightarrow \text{<character string lit>} \mid \text{<numeric lit>} \\
\text{<character string lit>} 
\Rightarrow \text{'}<\text{nonquote character}>\text{'} \\
\text{<nonquote character>} 
\Rightarrow \text{<numeric>} \\
\text{<newline>} \text{\textsuperscript{b}} 
\Rightarrow \text{<exact numeric lit>} \mid \text{<approx numeric lit>} \\
\text{<exact numeric lit>} 
\Rightarrow [ + \mid - ] \text{<unsigned int>} \mid [ . ] \text{<unsigned int>} \\
\text{<approx numeric lit>} 
\Rightarrow \text{<exact numeric lit>} \text{E} [ + \mid - ] \text{<digit>} \star \\
\text{<unsigned int>} 
\Rightarrow \text{<digit>} \star \\
\text{<correl name>} 
\Rightarrow \text{<ident>} \\
\text{<table name>} 
\Rightarrow \text{<ident>} \\
\text{<column>} 
\Rightarrow [ \text{<table name>} \cdot \mid \text{<correl name>} \cdot ] \text{<ident>} \\
\text{<table reference>} 
\Rightarrow \text{<table name>} [ \text{<correl name>} ] \\
\text{<value spec>} 
\Rightarrow \text{<ident>} \mid \text{<literal>} \\
\text{<comparison>} 
\Rightarrow = \mid < \mid > \mid <= \mid >= \\ \\
\text{\textsuperscript{a}}\text{A <nonquote character> is any character except the single quote character ('). This is as specified by the ANSI SQL Standard.} \\
\text{\textsuperscript{b}}\text{The ANSI SQL Standard states that the <newline> character is implementation dependent. For this implementation, the <newline> character is the linefeed character.} \\
\end{array}
\]
Semantic Definition of the SQL select Statement

### Table 2: Query Statement Syntax (adapted from the ANSI SQL Standard select statement syntax)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;query&gt;</code></td>
<td><code>select &lt;select list&gt; &lt;table expr&gt;</code></td>
</tr>
<tr>
<td><code>&lt;select list&gt;</code></td>
<td>`select &lt;query&gt; [ union</td>
</tr>
<tr>
<td><code>&lt;expn&gt;</code></td>
<td>`&lt;term&gt;</td>
</tr>
<tr>
<td><code>&lt;term&gt;</code></td>
<td>`&lt;factor&gt;</td>
</tr>
<tr>
<td><code>&lt;factor&gt;</code></td>
<td>`[ +</td>
</tr>
<tr>
<td><code>&lt;primary&gt;</code></td>
<td>`&lt;value spec&gt;</td>
</tr>
<tr>
<td><code>&lt;table expr&gt;</code></td>
<td><code>from &lt;table reference&gt; [ &lt;where&gt; ]</code></td>
</tr>
<tr>
<td><code>&lt;from&gt;</code></td>
<td><code>from &lt;table reference&gt; [ &lt;where&gt; ]</code></td>
</tr>
<tr>
<td><code>&lt;where&gt;</code></td>
<td><code>&lt;search cond&gt;</code></td>
</tr>
<tr>
<td><code>&lt;search cond&gt;</code></td>
<td>`&lt;bool term&gt;</td>
</tr>
<tr>
<td><code>&lt;bool term&gt;</code></td>
<td>`&lt;bool factor&gt;</td>
</tr>
<tr>
<td><code>&lt;bool factor&gt;</code></td>
<td><code>[ not ] &lt;bool primary&gt;</code></td>
</tr>
<tr>
<td><code>&lt;bool primary&gt;</code></td>
<td>`&lt;predicate&gt;</td>
</tr>
<tr>
<td><code>&lt;predicate&gt;</code></td>
<td><code>&lt;expn&gt; &lt;comparison&gt; &lt;expn&gt;</code></td>
</tr>
</tbody>
</table>

### 2.3 Notation

The notation used in Table 1 and Table 2 is BNF (Backus Normal Form) with the following extensions:

1. Square brackets ([...]) indicate optional elements.
2. An element $E$ repeated zero or more times is denoted as $E^*$. An element $E$ repeated one or more times is denoted as $E^+$.
3. Curly braces ({...}) indicate sequences of elements.

### 3 Semantic Definition

#### 3.1 Semantic Techniques

Standard semantic techniques using search and backtrack elements similar to Prolog were proposed for this project. Further study of the semantics of the SQL select statement indicated that the queries specified by the select statement were more readily described in terms of relational algebra. The relational algebra is described in terms of set operations (union, intersection, complex product) and several relational operations (selection, projection, division) [4].

For example, a typical SQL query using the select statement has the following form:
This query has a straightforward translation to relational algebra and set operations:

\[ \pi_{\text{select list}} (\sigma_{\text{search cond}} (\times (\text{<table list>}))) \]

where \( \pi \) is projection, \( \sigma \) is selection, and \( \times \) is the complex product. These operations will be defined more formally later.

With this in mind, it seems natural to define the semantics of SQL query statements in terms of these operations. This approach is desirable for several reasons. First, it makes the semantics more understandable because the meaning is stated in the well-defined terms of relational algebra. In addition, proving statements about the SQL query language should prove to be easier because proofs can draw upon a sound theoretical background and a large body of research on relational algebra. Finally, in this approach several abstract data types will be defined (relations, tuples, schemes). Abstracting these elements of the language allows the semanticist to prove statements about the abstract data types separate from the semantics of the language.

### 3.2 Informal Semantics

For the purposes of demonstrating the semantics of SQL queries, suppose there exists a database of hosts (computer systems) and processes. Also suppose there is a hosts relation and a process relation. Portions of these relations are shown in Figures 3 and 4.

A query such as: "select name, load1, status from host" would provide a table with 3 columns: name, load1, and status. The where clause allows qualification of the rows in the resulting table. For example,

\[
\text{select name, status, load1} \\
\text{from host} \\
\text{where load1 > 1.0 and name <> 'cs'}
\]

would produce Table 5, a 3-column table with all hosts whose load1 attribute is greater than 1.0 and whose name is not "cs".

<query>. A <query> operates on tables and produces a (possibly empty) table as a result. More complex queries can be formed by using the union or intersection operators. The union operator takes 2 tables, \( T_1 \) and \( T_2 \), and produces a new table containing the rows of both \( T_1 \) and \( T_2 \). Duplicate rows in the resulting table are removed. The intersection operator takes 2 tables, \( T_1 \) and \( T_2 \), and produces a new table containing the rows that are in both \( T_1 \) and \( T_2 \).
Semantic Definition of the SQL select Statement

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Users</th>
<th>Load1</th>
<th>Load5</th>
<th>Load15</th>
<th>Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>acme</td>
<td>down</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>()</td>
</tr>
<tr>
<td>asylum</td>
<td>up</td>
<td>5</td>
<td>1.24</td>
<td>1.24</td>
<td>1.27</td>
<td>(16107,19811,20170)</td>
</tr>
<tr>
<td>cs</td>
<td>up</td>
<td>25</td>
<td>1.34</td>
<td>1.00</td>
<td>0.92</td>
<td>(9834,12111,12112)</td>
</tr>
<tr>
<td>hellgate</td>
<td>up</td>
<td>0</td>
<td>1.70</td>
<td>0.92</td>
<td>0.70</td>
<td>(4112)</td>
</tr>
<tr>
<td>peruvian</td>
<td>up</td>
<td>28</td>
<td>1.63</td>
<td>1.11</td>
<td>1.30</td>
<td>(25,28,31,100,298)</td>
</tr>
<tr>
<td>jaguar</td>
<td>up</td>
<td>3</td>
<td>0.14</td>
<td>0.26</td>
<td>0.34</td>
<td>(2178)</td>
</tr>
<tr>
<td>jensen</td>
<td>up</td>
<td>1</td>
<td>0.43</td>
<td>0.13</td>
<td>0.04</td>
<td>(431,455)</td>
</tr>
<tr>
<td>shafer</td>
<td>up</td>
<td>1</td>
<td>0.09</td>
<td>0.10</td>
<td>0.04</td>
<td>(19928,19929,20050)</td>
</tr>
</tbody>
</table>

Table 3: Host Relation

<table>
<thead>
<tr>
<th>Name</th>
<th>PID</th>
<th>PPID</th>
<th>User</th>
<th>Mem</th>
<th>CPU</th>
<th>Size</th>
<th>RSS</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>rshd</td>
<td>25</td>
<td>298</td>
<td>root</td>
<td>0.3</td>
<td>0.0</td>
<td>100</td>
<td>60</td>
<td>peruvian</td>
</tr>
<tr>
<td>csh</td>
<td>28</td>
<td>25</td>
<td>hoogen</td>
<td>0.3</td>
<td>20.0</td>
<td>176</td>
<td>80</td>
<td>peruvian</td>
</tr>
<tr>
<td>ps</td>
<td>31</td>
<td>28</td>
<td>hoogen</td>
<td>0.9</td>
<td>18.9</td>
<td>252</td>
<td>228</td>
<td>peruvian</td>
</tr>
<tr>
<td>emacs</td>
<td>2178</td>
<td>2177</td>
<td>allen</td>
<td>2.5</td>
<td>0.6</td>
<td>996</td>
<td>700</td>
<td>jaguar</td>
</tr>
<tr>
<td>mail</td>
<td>431</td>
<td>430</td>
<td>cruse</td>
<td>0.0</td>
<td>0.0</td>
<td>208</td>
<td>36</td>
<td>jensen</td>
</tr>
<tr>
<td>xterm</td>
<td>455</td>
<td>454</td>
<td>yih</td>
<td>2.1</td>
<td>0.0</td>
<td>196</td>
<td>112</td>
<td>jensen</td>
</tr>
<tr>
<td>xcalc</td>
<td>12112</td>
<td>12111</td>
<td>zeleznik</td>
<td>0.8</td>
<td>0.0</td>
<td>208</td>
<td>204</td>
<td>cs</td>
</tr>
<tr>
<td>csh</td>
<td>9834</td>
<td>212</td>
<td>starkey</td>
<td>0.4</td>
<td>0.0</td>
<td>184</td>
<td>92</td>
<td>cs</td>
</tr>
</tbody>
</table>

Table 4: Process Relation

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Load1</th>
</tr>
</thead>
<tbody>
<tr>
<td>asylum</td>
<td>up</td>
<td>1.24</td>
</tr>
<tr>
<td>hellgate</td>
<td>up</td>
<td>1.70</td>
</tr>
<tr>
<td>peruvian</td>
<td>up</td>
<td>1.63</td>
</tr>
</tbody>
</table>

Table 5: Result of query 1.
The columns of the tables $T_1$ and $T_2$ must be of the same types. Thus, the $i^{th}$ column named in the query that produced $T_1$ must be the same type as the $i^{th}$ column named in the query that produced $T_2$.

**<select list>**. The **<select list>** is used to select specific attributes (columns) from tables. If all columns of a table are desired, the * character may be specified. Otherwise, all columns to be in the result are specified separated by commas. The columns named in the **<select list>** must unambiguously identify the desired column. If 2 tables in a query have the same column names, the column name must be prepended with a **<correl name>** to unambiguously identify the column. For example, this query involves the name column of the host and process tables:

```sql
select H.name, P.name from host H, process P
where H.name = 'asylum' and P.name = 'csh'
```

**<from>**. The **<from>** is used to specify the tables to be considered in the query. If the query involves comparing 2 or more instances of the same table, then a **<correl name>** can be specified to uniquely identify each instance. The **<correl name>** acts as a temporary and local name for the table. For example,

```sql
select C.pid, P.pid, P.name
from process P, process C
where P.pid = C.ppid and C.name = 'emacs'
```

will locate all processes with the name 'emacs'. A new table will be constructed containing the name of each of these processes, the process id, and name of their parent processes.

**<where>, <search cond>, and <predicate>**. The **<where>** clause is used to specify relationships that must exist among the columns in the result. The relationships are specified as a **<search cond>**. For a given row that is being considered by a query, the row is said to qualify if the **<search cond>** of the **<where>** clause evaluates to a truth value of true. Otherwise, the row is disqualified.

The **<predicate>** and **<search cond>** clauses allow the user to specify the conditions that qualify or disqualify a row from the result of a query. These conditions are evaluated to a truth value. For a given row, a true value will qualify the row for the resulting table. A false or unknown value will disqualify the row. Two or more **<predicate>** clauses can be connected by the boolean operators and and or to form more complex **<search cond>** clauses. The **<predicate>** clause is used to specify a comparison between values.

**<comparison>**. The comparison operators are the binary operators that appear in many programming languages. Numeric and character string values can be compared.
Comparison of numeric values is according to their algebraic values. Comparison of character strings is accomplished by comparing characters in the same ordinal position. If the strings are of different lengths, then the comparison is performed with the shorter string extended with spaces to the length of the longer string.

If a <predicate> is of the form “a <comparison> b”, then the result of <predicate> is either true, false, or unknown. The result is unknown if either a or b is null. The SQL Standard specifies that nothing, not even another null value, should be equal to the null value.

3.3 The Relational Model

The relational model of databases was first described by Codd [3] in 1970. Since then, it has been extensively studied. This section provides a brief overview of the relational model. Formal terminology and concepts introduced here are from Yang [9] and Delobel and Adiba [5] and will be used in subsequent sections.

Attributes

A relational database is composed of a set of attributes $A_1, A_2, \ldots, A_n$. Each attribute is composed of a single data type. The domain or value-set of each attribute $A_j$ is written $DOM(A_j)$. If $S$ is a subset of all possible attributes, we write $DOM(S) = \bigcup_{A_k \in S} A_k$.

Tuples

The rows in a relation are referred to as tuples. Formally, let $X$ be a subset of the set of all attributes in the database. Let $DOM(X)$ be the domain of $X$. Let $\mu: X \rightarrow DOM(X)$ be a function defined as follows:

$$\mu = \{(A_{i_1}, a_1), \ldots, (A_{i_k}, a_k)\}$$

Each $A_{i_j}$ for $1 \leq j \leq k$ is an attribute in $X$ and an argument to the function $\mu$. Each $a_j$ for $1 \leq j \leq k$ is a value in $DOM(A_{i_j})$. Thus, $\mu(A_{i_j}) = a_j$. The function $\mu$ is called a tuple over $X$.

If $Y \subseteq X$, then the $Y$-value(s) of $\mu$ is written

$$\mu[Y] = \{(A_m, a_m) \mid a_m = \mu[A_m], 1 \leq m \leq |Y|, A_m \in X\}$$

All possible tuples over $X$ is the complex product of each domain $A_{i_j}$. The complex product of domains is defined as:

$$TUP(X) = \times(DOM(A_{i_1}), DOM(A_{i_2}), \ldots, DOM(A_{i_k}))$$

$$= \{a_1a_2\ldots a_k \mid a_j \in DOM(A_{i_j}), j = 1, 2, \ldots, k\}$$

Thus, each tuple $\mu$ over $X$ is an element of $TUP(X)$. 
Relation Schemes

The scheme of a relation is also known as the intention of the relation. The scheme of a relation is defined as a subset of all possible attributes of the database. Thus, a relation over S is a subset of $TUP(S)$. The cardinality of a scheme is the number of attributes in the scheme.

Relations

A relation is defined in terms of its scheme. Thus, a relation over S is a relation with scheme S. The arity of a relation over S is equal to the cardinality of S. The cardinality of a relation is equal to the number of tuples belonging to that relation.

Since the relation domain is the key component of the query language, several important operations on relations are discussed here.

Union. The union of two relations $R_1$ and $R_2$ is a relation $R_3$ whose tuples are also in $R_1$ or $R_2$ (or both). Formally,

$$R_1 \cup R_2 = \{ \mu | \mu \in R_1 \text{ or } \mu \in R_2 \}$$

This operation is allowed only when $R_1$ and $R_2$ are union-compatible. Union-compatibility is defined as the following property:

**Definition 1** Two relations $R_1$ and $R_2$ are union-compatible if their schemes are union-compatible.

**Definition 2** Two Schemes A and B are union-compatible if there exists a bijection such that for each $A_j \in A$ there exists exactly one $B_k \in B$ for which $DOM(A_j) = DOM(B_k)$.

Note that the union of any relation R and the empty relation $\Phi$ is the relation R.

$$R \cup \Phi = R$$

Complex Product. The complex product is very similar to the cartesian product except the ordering of the attributes in a complex product is insignificant. The cartesian product of two relations $R_1$ and $R_2$ is denoted $R_1 \times R_2$ and produces all possible pairs of tuples $(\mu_{R_1}, \mu_{R_2})$. Rather than a pair of tuples, the complex product produces the concatenation of the tuples.

The complex product of two relations $R_1$ and $R_2$ is denoted $R_1 \ast R_2$ and is defined as

$$R_1 \ast R_2 = \{ \mu | \mu_{R_1}, \mu_{R_2} \text{ and } (\mu_{R_1}, \mu_{R_2}) \in R_1 \times R_2 \}$$
Intersection. The intersection of two relations $R_1$ and $R_2$ is a relation $T$ whose tuples are in $R_1$ and $R_2$. Formally,

$$R_1 \cap R_2 = \{ \mu | \mu \in R_1 \text{ and } \mu \in R_2 \}$$

This operation is allowed only when $R_1$ and $R_2$ are union-compatible. Note that the intersection of any relation $R$ and the empty relation $\Phi$ is the empty relation.

$$R \cap \Phi = \Phi$$

Projection. The projection operation takes a relation and produces a relation whose attributes are a subset of the original relation’s attributes. Formally, if $R$ is a relation over $S$ and $X$ is a subset of $S$, then the projection of $R$ onto $X$, written as $\pi_X(R)$ is a relation over $X$.

$$\pi_X(R) = \{ \mu[X] | X \subseteq S \text{ and } \mu \in R \}$$

Selection. The selection operation allows certain tuples to be selected from a relation. The condition for selection is specified in a formula which is defined inductively as [9]:

1. $F_1 \Omega F_2$, $F_1 \Omega c$, and $c \Omega F_1$ are formulas where $F_1$ and $F_2$ are compatible attributes, $c$ is a constant in $\text{DOM}(F_1)$, and $\Omega$ is an arithmetic operator in $\{=, \neq, <, \leq, >, \geq \}$.
2. If $F_1$ and $F_2$ are formulas, then $F_1 \text{ and } F_2$, $F_1 \text{ or } F_2$, $\text{not } F_1$, and $\text{not } F_2$ are formulas.
3. Nothing else is a formula.

The selection of a relation $R$ under a formula $F$ is a subset of $R$ consisting of all the tuples of $R$ that satisfy $F$. It is written as:

$$\sigma_F(R) = \{ \mu | \mu \in R \text{ and } \mu \text{ satisfies } F \}$$

If $F$ is the null formula, then $\sigma_F(R) = R$.

3.4 Formal Semantics

Semantic Algebras

Primitive domains to be used are stated here without further explanation. These domains are as stated in Schmidt [8].

Integer numbers. $i \in \mathbb{Z} = \text{Int}$

Floating Point numbers. $v \in \mathbb{R} = \text{Real}$
Semantic Definition of the SQL select Statement

Character Strings. \( s \in C = \text{String} \)

Boolean Values. \( s \in \text{Tr} = \text{Tr} \)

Identifiers. \( id \in \text{Id} = \text{Identifier} \)

Lists. \( L \in D^* \). As specified in Example 3.9, pp 43–44 of Schmidt [8].

DenotableValue. \( dv \in \text{DenotableValue} = \text{StorableValue} + \text{Identifier} + \text{Error} \)

StorableValue. \( sv \in \text{StorableValue} = \text{NULL} + \text{Int} + \text{Real} + \text{String} + \text{Tr} + \text{Error} \)

Store. \( st \in \text{Store} = \text{Identifier} \rightarrow \text{Relation} \)

This is similar to that specified in Figure 7.1, pg. 140 of Schmidt except the only data type that is storable is a relation.

\[
\text{newstore} : \text{Store} \\
\text{newstore} = \lambda i. \text{error}
\]

\[
\text{access} : \text{Identifier} \rightarrow \text{Store} \rightarrow \text{Store} \\
\text{access} = \lambda i. \lambda r. \lambda s(i)
\]

\[
\text{update} : \text{Identifier} \rightarrow \text{Relation} \rightarrow \text{Store} \rightarrow \text{Store} \\
\text{update} = \lambda i. \lambda r. \lambda s(\lambda i'. i \text{ equals } i' \rightarrow r[|s(i')]\)
\]

Answer. \( a \in \text{Answer} = (\text{Relation} \star \text{State}) + \text{Error} \)

State. \( state \in \text{State} = \text{Store} \)

Query Continuation. \( qc \in Q\text{Cont} = \text{Relation} \star \text{State} \rightarrow \text{Answer} \)

Expression List Continuation.
\( elc \in E\text{LCont} = \text{ExpList} \star \text{Relation} \star \text{State} \rightarrow \text{Answer} \)

Expression List. \( el \in \text{ExpList} = \text{Expression} + (\text{Expression} \star \text{ExpList}) \)

Type. \( t \in \text{Type} = \text{Int} + \text{Real} + \text{String} + \text{Tr} + \text{Null} + \text{TypeError} \)
Abstract Data Types

**Scheme.** \( s \in \text{Scheme} = (\text{Identifier} \times \text{Type} \times \text{Scheme}) + \text{EmptyScheme} + \text{SchemeError} \)

**createScheme**: Scheme 
- \( \text{createScheme} = \text{EmptyScheme} \)

**addToScheme**: Scheme \(\rightarrow\) Identifier \(\rightarrow\) Type \(\rightarrow\) Scheme 
- \( \text{addToScheme} = \lambda s.\lambda i.\lambda t.(i, t, s) \)

**idTypeInScheme**: Scheme \(\rightarrow\) Identifier \(\rightarrow\) Type
- \( \text{idTypeInScheme} = \lambda(i, t, s).\lambda i'.(i \text{ equals } i') \rightarrow t \)
  
  - \( (s \text{ equals EmptyScheme}) \rightarrow \text{TypeError} \)
  
  - \( \text{idTypeInScheme s } i' \)

**concatScheme**: Scheme \(\rightarrow\) Scheme \(\rightarrow\) Scheme
- \( \text{concatScheme} = \lambda(i, t, s).\lambda(i', t', s').((\text{eqScheme } s \text{ EmptyScheme}) \text{ and } (\text{eqScheme } s' \text{ EmptyScheme})) \)
  
  - \( (i, t, (i', t', \text{EmptyScheme})) \rightarrow (i, t, (i', t', s')) \)
  
  - \( (i, t, (\text{concatScheme } s (i', t', s'))) \)

**eqScheme**: Scheme \(\rightarrow\) Scheme \(\rightarrow\) Tr
- \( \text{eqScheme} = \lambda(i, t, s).\lambda(i', t', s').((i \text{ equals } i') \text{ and } (t \text{ equals } t') \text{ and } (s \text{ equals EmptyScheme}) \text{ and } (s' \text{ equals EmptyScheme})) \rightarrow \text{true} \)
  
  - \( ((i \text{ equals } i') \text{ and } (t \text{ equals } t') \text{ and } (\text{eqScheme } s s')) \rightarrow \text{true} \)
  
  - \( \text{false} \)

**unionCompatible**: Scheme \(\rightarrow\) Scheme \(\rightarrow\) Tr
- \( \text{unionCompatible} = \lambda(i, t, s).\lambda(i', t', s').((t \text{ equals } t') \text{ and } (\text{eqScheme } s \text{ EmptyScheme}) \text{ and } (\text{eqScheme } s' \text{ EmptyScheme})) \rightarrow \text{true} \)
  
  - \( (((t \text{ equals } t') \text{ and } (\text{unionCompatible } s s')) \rightarrow \text{true} \)
  
  - \( \text{false} \)

**cardinality**: Scheme \(\rightarrow\) Int
Semantic Definition of the SQL select Statement

\[
\text{cardinality} = \lambda(i,t,s).\text{seqScheme EmptyScheme} \rightarrow 0 \| 1 + \text{cardinality}(s)
\]

**Tuple.** \( t \in \text{Tuple} = (\text{Identifier} \times \text{StorableValue} \times \text{Tuple}) + \text{EmptyTuple} + \text{TupleError} \)

\[
\text{createTuple} : \text{Tuple} = \text{EmptyTuple}
\]

\[
\text{addJoTuple} : \text{Tuple} ightarrow \text{Identifier} ightarrow \text{Tuple} = \lambda t.\lambda i.(i,\text{inStorableValue}(\text{NULL}),t)
\]

\[
\text{updateTuple} : \text{Identifier} ightarrow \text{StorableValue} ightarrow \text{Tuple} ightarrow \text{Tuple} = \lambda i'.\lambda s v'.\lambda (i,s,v,t).(i \text{ equals } i') \rightarrow (i,s v',t) \rightarrow (\text{eqTuple} t \text{ EmptyTuple}) \rightarrow \text{tupleError} \rightarrow (\text{updateTuple} i' s v' t)
\]

\[
\text{concatTuple} : \text{Tuple} ightarrow \text{Tuple} ightarrow \text{Tuple} = \lambda (i,s,v,t).\lambda (i',s v',t').((\text{eqTuple} t \text{ EmptyTuple}) \text{ and} (\text{eqTuple} t' \text{ EmptyTuple}) \rightarrow \lambda (i,s,v,t1).(i,sv,t) \rightarrow (i',sv',t') \rightarrow (\text{concatTuple} t (i,sv,t1)) \rightarrow \lambda (i',sv',t').((\text{eqTuple} t' \text{ EmptyTuple}) \rightarrow \lambda (i',sv',t').((i,sv,t1) \rightarrow (i',sv',t')) \rightarrow \text{true} \rightarrow \text{false}
\]

\[
\text{eqTuple} : \text{Tuple} ightarrow \text{Tuple} \rightarrow \text{Tr} = \lambda (i,s,v,t).\lambda (i',sv',t').((i \text{ equals } i') \text{ and} (sv \text{ equals } sv') \text{ and} (t \text{ equals } \text{EmptyTuple}) \text{ and} (t' \text{ equals } \text{EmptyTuple}) \rightarrow \text{true} \rightarrow (i',sv',t') \rightarrow \text{true} \rightarrow \text{false}
\]

\[
\text{arity} : \text{Tuple} \rightarrow \text{Int} = \lambda (i,s,v,t).(t \text{ eqTuple EmptyTuple}) \rightarrow 0 \| 1 + \text{arity}(t)
\]

**Relation.** \( r \in \text{Relation} = (\text{Scheme} \times \text{Tuple list}) + \text{RelationError} \)

\[
\text{createRelation} : \text{Relation} = (\text{createScheme()},[])
\]

\[
\text{getRelationScheme} : \text{Relation} \rightarrow \text{Scheme} = \lambda (s,tupleList).s
\]
getRelationTuples : Relation → Tuplelist
  getRelationTuples = \( s, \text{tupleList} \). \text{tupleList}

arity : Relation → Int
  arity = \( \lambda r. (\text{cardinality}(\text{getRelationSchemer}(r))) \)

cardinality : Relation → Int
  cardinality = \( \lambda r. (\text{length}(\text{getRelationTuples}(r))) \)

updateRelationScheme : Relation → Scheme → Relation
  updateRelationScheme = \( s, tL \). \( s', tL \)

addTupleToRelation : Tuple → Relation → Relation
  addTupleToRelation = \( t, s, tL \). \( s, t \text{ cons} \ tL \)

memberOfRelation : Tuple → Relation → Tr
  memberOfRelation = \( t, s, tL \). \text{null}(tL) → false
  \[ (\text{eqTuple}(t, \text{hd}(tL))) → true \]
  \[ (\text{memberOfRelation}(t, tL)) \]

intersection : Relation → Relation → Relation
  intersection = \( \lambda r, r'. (\text{intersection}\_\text{aux}(\text{createRelation}()) \ r \ r') \)

intersection\_aux : Relation → Relation → Relation → Relation
  intersection\_aux =
  \( \lambda r, s, tL, s', tL'. (tL'\text{equals}\_[]() \rightarrow rslt \]
  \[ (tL'\text{equals}\_[]()) \rightarrow rslt \]
  \[ (\text{memberOfRelation}(\text{hd}(tL), r')) \rightarrow (\text{intersection}\_\text{aux} \]
  \[ (\text{addTupleToRelation}(\text{hd}(tL), rslt), \]
  \[ (s, tL((tL))) \]
  \[ (s', tL')) \]
  \[ (\text{intersection}\_\text{aux} \ rslt(s, tL((tL))(s', tL')) \]

selection : Relation → Expression → Relation
  selection =
  \( \lambda r, \text{e} . (\text{selection}\_\text{aux} \]
  \[ (\text{updateRelationScheme}(\text{createRelation}()) (\text{getRelationScheme}(r))) \]
  \[ r \]
  \[ \text{e} \)
Semantic Definition of the SQL `select` Statement

```
selection_aux : Relation → Relation → Expression → Relation
selection_aux =
λrslt.λ(s,tList).λe.(tList'equals[]) → rslt
  (let r = E[e] hd(tList)
    in (cases r of
      isBoolean(tv) →
        (tv → (selection_aux
          (addTupleToRelation
            hd(tList) rslt)
          (s, tl(tList))
          e)
        rslt[]) (let r = E[e] hd(tList)
          in (cases r of
            isBoolean(tv) →
              (tv → (selection_aux rslt (s, tl(tList)) e))
              [] RelationError)
          end))

The remaining operations on relations are similar or composed of the operations
shown above. The union and complex product operations are very similar to intersection
and projection is very similar to selection.

3.5 Valuation Functions

Q:Query → QCont → Answer

Q[Θ qop Ψ] qc =
S[Θ] λ (r,s).(cases qop of
  "union" →
    (cases r of
      (...) → (let qc' = Q[Ψ] λ (r',s').
        (cases r' of
          (...) → qc ((union r r'),s')
        )
        RelationError → Error)
    (cases r of
      (...) → (let qc' = Q[Ψ] λ (r',s').
        (cases r' of
          (...) → qc ((intersection r r'),s')
        )
        RelationError → Error)
    RelationError → Error)
  )

Q[Θ] qc = S[Θ] qc

S:Statement → QCont → QCont
```
Semantic Definition of the SQL select Statement

\[ S[* \Gamma] \text{qc} = \lambda (r',s').(\text{let} \text{qc}' = \text{TE}[\Gamma] \text{qc} \text{in} \text{qc}' (r',s') \text{end}) \]

\[ S[\varepsilon,\Sigma \Gamma] \text{qc} = \]

\[ \text{TE}[\Gamma] \text{Sel}[\varepsilon,\Sigma] \lambda (eL,r,s). \text{qc} ((\text{projection} r \ eL),s) \]

Sel:Selection \rightarrow \text{ELCont} \rightarrow \text{QCont}

\[ \text{Sel}[\varepsilon] \text{elc} = \lambda (r,s).\text{elc} (\varepsilon,r,s) \]

\[ \text{Sel}[\varepsilon,\Sigma] \text{elc} = \]

\[ \text{Sel}[\Sigma] \lambda (eL,r,s).\text{elc} ((\varepsilon,eL),r,s) \]

\[ \text{TE:TableExp} \rightarrow \text{QCont} \rightarrow \text{QCont} \]

\[ \text{TE[from} T \text{] qc} = \text{TL}[T] \text{qc} \]

\[ \text{TE[from} T \text{ where} \varepsilon \text{] qc} = \]

\[ \text{TL}[T] \lambda (r,s).((\text{cases} \ r \ \text{of} \quad (\ldots) \rightarrow \text{qc} ((\text{selection} r \ \varepsilon),s) \quad \| \text{Error}) \]

\[ \text{TL:TableList} \rightarrow \text{QCont} \rightarrow \text{QCont} \]

\[ \text{TL}[\varepsilon] \text{qc} = \text{T}[\varepsilon] \lambda (r,s).\text{qc} (r,s) \]

\[ \text{TL}[\varepsilon,\Gamma] \text{qc} = \]

\[ \text{T}[\varepsilon] \lambda (r,s).((\text{let} \text{qc}' = \text{TL}[\Gamma] \text{qc} \text{in} \text{qc}' (r,s) \text{end}) \]

\[ \text{T:Table} \rightarrow \text{QCont} \rightarrow \text{QCont} \]

\[ \text{T}[\varepsilon] \text{qc} = \]

\[ \lambda (r,\text{store}).((\text{cases} \ r \ \text{of} \quad (\ldots) \rightarrow \text{qc} ((\text{cartesianProduct} r \ r'),\text{store}) \quad \| \text{RelationError} \rightarrow \text{Error}) \]

\[ \text{RelationError} \rightarrow \text{Error}) \]

\[ \text{T}[\varepsilon,\xi'] \text{qc} = \]

\[ \lambda (r,\text{store}).((\text{cases} \ r \ \text{of} \quad (\ldots) \rightarrow \text{qc} ((\text{cartesianProduct} r \ r'),\text{store}) \quad \| \text{RelationError} \rightarrow \text{Error}) \]
Expressions are evaluated only with respect to a particular tuple of a relation. The tuple (and its Scheme) provide all necessary information in order to evaluate an expression. The expression valuation function evaluates the expression to its normal form.

\[
E: \text{Expression} \rightarrow \text{Tuple} \rightarrow \text{Scheme} \rightarrow \text{Expression}
\]

\[
E[\varepsilon_1 + \varepsilon_2] t s =
\]

\[
(\text{let } \varepsilon_1' = E[\varepsilon_1] t s
\]

\[
\text{in } (\text{let } val \varepsilon_2' = E[\varepsilon_1] t s
\]

\[
\text{in } (\text{cases } \varepsilon_1' \text{ of }
\]

\[
\text{isColumn}(\text{col}) \rightarrow
\]

\[
(cases \varepsilon_2' \text{ of }
\]

\[
\text{isColumn}(\text{col}') \rightarrow
\]

\[
((\text{id}._\text{type}_\text{inScheme} s C[\text{col}])
\]

\[
= \text{INT})
\]

\[
\rightarrow
\]

\[
\text{inExpression}((\text{accessTuple} t C[\text{col}]) + i)
\]

\[
\text{isReal}(v) \rightarrow
\]

\[
((\text{id}._\text{type}_\text{inScheme} s C[\text{col}])
\]

\[
= \text{REAL})
\]

\[
\rightarrow
\]

\[
\text{inExpression}((\text{accessTuple} t C[\text{col}]) + v))
\]

\[
\text{isInt}(i) \rightarrow
\]

\[
(cases \varepsilon_2' \text{ of }
\]

\[
\text{isColumn}(\text{col}') \rightarrow
\]

\[
(\text{INT}
\]

\[
= \text{INT}_\text{col'}))
\]
Other operations (boolean, relational, arithmetic) are completed in a similar fashion. The base cases in expression evaluation are as follows:

\[
E[S] \, t \, s = \text{inExpression}(S) \\
E[I] \, t \, s = \text{inExpression}(I) \\
E[R] \, t \, s = \text{inExpression}(R) \\
E[B] \, t \, s = \text{inExpression}(B) \\
E[NULL] \, t \, s = \text{inExpression}(\text{NULL}) \\
E[C] \, t \, s = \\
\quad \text{(let rslt = \text{accessTuple} \, t \, C[C])} \\
\quad \text{in (cases rslt of} \\
\quad \quad \text{isInt}(i) \, \rightarrow \, \text{inExpression}(i) \\
\quad \quad \text{[ExpError)} \\
\quad \text{end)} \\
\quad \text{[ExpError)} \\
\quad \text{isReal}(v) \, \rightarrow \\
\quad \quad \text{(cases \, \text{isReal}(v) \, \rightarrow \, \text{inExpression}(v) \\
\quad \quad \text{[ExpError)} \\
\quad \text{end)} \\
\quad \text{end)} \\
\text{end)}
\]
4 Equivalence of Queries

A given query can be expressed in more than one way. In particular, the union and intersection of two queries on the same relations has an equivalent single query form. Furthermore, the single query form frequently involves less typing.

Suppose \( \Psi_1, F_1 (R_1, R_2, \ldots, R_n) \) and \( \Psi_2, F_2 (R_1, R_2, \ldots, R_n) \) are two queries over relations \( R_1, R_2, \ldots, R_n \). Also, suppose these queries involve a selection operation using \( F_1 \) and \( F_2 \), respectively.

The union and intersection operations are only defined for relations that are union-compatible. Thus, \( \Psi_1, F_1 (R_1, R_2, \ldots, R_n) \) and \( \Psi_2, F_2 (R_1, R_2, \ldots, R_n) \) must be union-compatible.

**Union.** The union of these two queries is the relation containing the tuples that satisfy \( F_1 \) or \( F_2 \) (from the definition of union of two relations). The definition of a formula for selection says that if \( F_1 \) and \( F_2 \) are formulas, then \( F_1 \) and \( F_2 \) joined by the boolean or operation is also a formula. Let \( F_3 = F_1 \text{ or } F_2 \). The syntax for the select statement allows a query of the form \( \Psi_3, F_3 (R_1, R_2, \ldots, R_n) \).

**Intersection.** Showing that there exists a single-query form of the intersection of two union-compatible queries over the same relations is analogous to showing the same property for union.

5 Conclusions

The semantics of a portion of the SQL select statement has been described. The description demonstrates the combination of standard semantics style which was used to describe how an SQL query is transformed into a series of operations on relations, and abstract data types used to describe the relational database model. The resulting semantics are operational in style because the semantics of a query are described in terms of well-defined operations on the abstract data types. This makes correctness and other types of proofs easier because the correctness of the abstract data types can be dealt with separately.

In the process of defining the semantics of the SQL select statement, several observations were made:
Initially, the model developed for how a query "works" required the use of an environment (in a reference like host H, H would be a local variable) and store (to store relations). Further investigation into what a query actually does revealed a much simpler series of operations than was first anticipated. The resulting semantics do not require an environment. Expanding the syntax in any way to handle more of the SQL syntax would require the use of an environment, however. This could easily be done by making a \( \text{State} = \text{Environment} \ast \text{Store} \) rather than \( \text{State} = \text{Store} \) as it is now.

Portions of the chosen syntax are unnecessary. In particular, the union and intersection of two queries have equivalent single query forms involving the or and and boolean operations.

References


UUCS-92-002 Akella, Venkatesh and Ganesh Gopalakrishnan. "CFSIM: A Concurrent Compiled Code Simulator for hopCP." Xeroxing Cost: "$1.00 (20 pgs) **

UUCS-92-003 Cohen, Michael F. "Spacetime Control of Linked Figures." Xeroxing Cost: $5.85 (117 pgs) **


UUCS-91-001 Gershon, Elber and Elaine Cohen. "Error Bounded Variable Distance Offset Operator for Free From Curves and Surfaces." Xeroxing Cost: $.80 (16 pgs)

UUCS-91-002 Starkey, Mike. "A Lisp-based OCCAM Interpreter." Xeroxing Cost: $.65 (13 pgs)

UUCS-91-003 Starkey, Mike and T.M. Carter. "Transforming Disfigured And Disoriented Areas Into Routable Switchboxes." Xeroxing Cost: $.35 (7 pgs)

UUCS-91-004 Starkey, Mike and T.M. Carter. "Switchbox Routing By Pattern Matching." Xeroxing Cost: $.60 (12 pgs)

UUCS-91-005 Ming, John Charles. "A Multistrategy Machine Learning Approach To Object Recognition." Xeroxing Cost: $13.05 (261 pgs) *

UUCS-91-006 Sohrt, Wolfgang. "Interaction With Constraints In Three-Dimensional Modeling." Xeroxing Cost: $5.80 (116 pgs) *

UUCS-91-007 Susswein, Steven Y. "Parallel Path Consistency." Xeroxing Cost: $3.45 (69 pgs) *


Susswein, Steven Y., Henderson, Thomas C., Zachary, Joseph L., Hansen, Chuck, Hinker, Paul and Gary C. Marsden. "Parallel Path Consistency." Xeroxing Cost: $1.00 (20 pgs)


Akella, Venkatesh and Ganesh Gopalakrishnan. "Hierarchical Action Refinement: A Methodology For Compiling Asynchronous Circuits from a Concurrent HDL." Xeroxing Cost: $1.05 (21 pgs)


Smith, Jeffrey Kent. "SEQWARP: A Low-Cost Linear Systolic Array for Biological Sequence Comparison." Xeroxing Cost: $3.90 (78 pgs)


Bracha, Gilad and Gary Lindstrom. "Modularity Meets Inheritance." Xeroxing Cost: $.90 (18 pgs)


George, Lal and Gary Lindstrom. "Using a Functional Language And Graph Reduction To Program Multiprocessor Machines." Xeroxing Cost: $

Akella, Venkatesh and Ganesh Gopalakrishnan. "hopCP: A Concurrent Hardware Description Language." Xeroxing Cost: $1.70 (34 pgs)

Jain, Prabhat, Ganesh Gopalakrishnan and Prabhakar Kudva. "Verification of Regular Arrays by Symbolic Simulation." Xeroxing Cost: $.80 (16 pgs)


Hoogenboom, Peter. J.. "Semantic Definition of a Subset of the Structured Query Language (SQL)." Xeroxing Cost: $1.05 (21 pgs)

Short, Wolfgang. "Interaction With Constraints in Threee-Dimensional Modeling." Xeroxing Cost: $5.70 (11 pgs)
<p>| UUCS-91-029 | Swanson, Mark R.. &quot;Concurrent Scheme, A Language for Concurrent Symbolic Computing.&quot; Xeroxing Cost: $5.70 (114 pgs) |
| UUCS-91-031 | Susswein, Steven Y.. &quot;Parallel Path Consistency.&quot; Xeroxing Cost: $3.45 (69 pgs) |
| UUCS-91-032 | Paik, Karen Lynn. &quot;Trivariate B-Splines.&quot; Xeroxing Cost: $3.00 (60 pgs) |
| UUCS-90-005 | Bruderlin, Beat. &quot;DI-A Portable, Object Oriented User Interface Toolkit for Modlula-2 Applications.&quot; Xeroxing Cost: $2.10 (42 pgs) |
| UUCS-90-017 | Evans, John. &quot;A Distributed Object Oriented Graphical Programming System.&quot; Xeroxing Cost: $5.45 (109 pgs) |</p>
<table>
<thead>
<tr>
<th>Document Number</th>
<th>Title</th>
<th>Author(s)</th>
<th>Price</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>UUCS-90-020</td>
<td>&quot;DPOS Programming Manual.&quot;</td>
<td>Evans, John D.</td>
<td>$1.05</td>
<td>21</td>
</tr>
<tr>
<td>UUCS-90-021</td>
<td>&quot;Combining Verification and Simulation.&quot;</td>
<td>Gopalakrishnan, Jain, Akella, Josephson, Kuo.</td>
<td>$.85</td>
<td>17</td>
</tr>
<tr>
<td>UUCS-90-022</td>
<td>&quot;Elastic And Plastic Surfaces For Modeling And Animation.&quot;</td>
<td>Thingvold, Jeffrey Allyn</td>
<td>$5.80</td>
<td>116</td>
</tr>
<tr>
<td>UUCS-90-023</td>
<td>&quot;An Axiomatic Approach for Solving Geometric Problems Symbolically.&quot;</td>
<td>Bruderlin, Beat</td>
<td>$1.95</td>
<td>29</td>
</tr>
<tr>
<td>UUCS-90-024</td>
<td>&quot;Interaction With Constraints in 3D Modeling.&quot;</td>
<td>Bruderlin, Beat</td>
<td>$1.00</td>
<td>20</td>
</tr>
<tr>
<td>UUCS-90-025</td>
<td>&quot;Static Analysis of Multiparadigm Languages.&quot;</td>
<td>Yeh, Dowming.</td>
<td>$7.90</td>
<td>158</td>
</tr>
<tr>
<td>UUCS-90-026</td>
<td>&quot;Intersurface Continuity of B-Spline Based Solid Models.&quot;</td>
<td>Livingston, John Barry</td>
<td>$4.90</td>
<td>98</td>
</tr>
<tr>
<td>UUCS-90-027</td>
<td>&quot;Computer Illustration of Three Dimensional Sculptured Surfaces.&quot;</td>
<td>Dooley, Debra Lynn</td>
<td>$4.20</td>
<td>84</td>
</tr>
<tr>
<td>UUCS-89-001</td>
<td>&quot;Automatic Mesh Analysis Technique by Knowledge-Based System.&quot;</td>
<td>Lee, Hyo Jong.</td>
<td>$2.15</td>
<td>43</td>
</tr>
<tr>
<td>UUCS-89-003</td>
<td>&quot;An Abstract Machine for Parallel Graph Reduction.&quot;</td>
<td>George, Lal.</td>
<td>$1.00</td>
<td>20</td>
</tr>
<tr>
<td>UUCS-89-005</td>
<td>&quot;Cascade: Hardware for High/Variable Precision Arithmetic.&quot;</td>
<td>Carter, Tony.</td>
<td>$.60</td>
<td>12</td>
</tr>
<tr>
<td>UUCS-89-006</td>
<td>&quot;Cascade: A Hardware Alternative to Bignums.&quot;</td>
<td>Carter, Tony.</td>
<td>$1.35</td>
<td>27</td>
</tr>
<tr>
<td>UUCS-89-008</td>
<td>&quot;Structured, Technology Independent VLSI Design.&quot;</td>
<td>Gu, Jun and Smith, Kent</td>
<td>$2.35</td>
<td>47</td>
</tr>
<tr>
<td>UUCS-89-009</td>
<td>&quot;A Tour of the Worm.&quot;</td>
<td>Seeley, Donn.</td>
<td>$.75</td>
<td>15</td>
</tr>
<tr>
<td>UUCS-89-010</td>
<td>&quot;Automatic Sculptured Five-Axis Milling with Check Surfaces.&quot;</td>
<td>Chou, Jin J. and Cohen, Elaine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UUCS-89-011</td>
<td>&quot;Constant Scallop Height Tool Path Generation.&quot;</td>
<td>Chou, Jin J. and Cohen, Elaine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UUCS-89-012</td>
<td>&quot;Logical Behaviors.&quot;</td>
<td>Henderson, Tom.</td>
<td>$1.20</td>
<td>24</td>
</tr>
</tbody>
</table>


UUCS-89-016  Carter, Tony and Smith., Kent. "Path-Programmable Logic"


UUCS-89-026  "Ma, Kwan-Liu. "TICL: A Type Inference System for Common Lisp." Xeroxing Cost: $4.10 (82 pgs)


UUCS-88-002  Brandt, Richard C., "Building Databases for the Computer-Based Memorization System." Xeroxing Cost: $1.35 (27 pages)


UUCS-88-008  Bloomentahl, Mark "Approximation of Sweep Surfaces by Tensor Product B-splines."(cost 72 pages, $3.60)

UUCS-88-009  Cobb, James E., "Tiling the Sphere with Rational


Thompson, Richard. "Design Specifications for Hardware Assisted Rollback Computation." $0.75/15 pgs.


Shebs, Stanley T. "Implementing Primitive Datatypes from Higher-Level Languages." Xeroxing cost: $8.30/166 pgs.

Fujimoto, Richard M. "Time Warp on a Shared Memory Multiprocessor" Xeroxing Cost: $1.30 / 26 Pgs.

Gatrell, Lance B. "CAD-Based Grasp Synthesis" Xeroxing Cost: $2.55 / 51 pgs.

Gu, Jun and Smith, Kent F. "Fast Structured Design of VLSI Circuits." Xeroxing Cost: $2.45 (49 pgs)

Sosic, Rok and Gu, Jun. "How to search for Millions of Queens" Xeroxing Cost: $0.80 (16 pgs)


Ho, Chih-Cheng. "Computer-Aided Geometric Design Based Three-Dimensional Object Representations For Computer Vision." Xeroxing Cost: $3.95 (79 pgs)


Pourheidri, Mohammad. "Moped (A Portable Debugger)." Xeroxing Cost: $3.05 (61 pgs)

Elvins, Thomas Todd. "Modeling Systems of Rigid Bodies in Motion Using Dynamics."
<table>
<thead>
<tr>
<th>Document Code</th>
<th>Title</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UUCS-88-032</td>
<td>&quot;A Multiprocessor Implementation of CSP.&quot;</td>
<td>Feng, Hwa-chung</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $5.40 (108 pgs)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-002</td>
<td>&quot;PPL Quick Reference Guide (CMOS).&quot;</td>
<td>Jacobs, Steven R. and Smith, Kent F.</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $2.70 (54 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-004</td>
<td>&quot;Set Operation on Sculptured Solids&quot;</td>
<td>Thomas, Spencer W.</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $1.55 (31 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-006</td>
<td>&quot;Workshop on Multisensor Integration in Manufacturing Automation.&quot;</td>
<td>Henderson, Thomas C., Allen, Peter, Cox, Ingemar, Mitiche, Amar,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Durrant-Whyte, Hugh, and Snyder, Wes.</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $1.65 (33 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-007</td>
<td>&quot;CAOS An Approach to Robot Control.&quot;</td>
<td>Thune, Nils and Bhanu, Bir</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $2.30 (46 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-008</td>
<td>&quot;Implementation of a Hierarchical Control System on a BBN Butterfly</td>
<td>Thune, Mari and Bhanu, Bir</td>
</tr>
<tr>
<td></td>
<td>Multiprocessor: Initial Studies and Results.&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $3.70 (74 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-010</td>
<td>&quot;High-Level Planning for Dextrous Manipulation.&quot;</td>
<td>Grupen, Roderic A. and Henderson, Thomas C.</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $1.15 (13 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-011</td>
<td>&quot;Apparent Symmetries in Range Data.&quot;</td>
<td>Grupen, Roderic A. and Henderson, Thomas C.</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $0.45 (9 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-012</td>
<td>&quot;Managing Large Address Spaces Effectively on the Butterfly.&quot;</td>
<td>Tinker, Peter</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $0.55 (11 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-013</td>
<td>&quot;An Approach to Three-Dimensional Scene Databases.&quot;</td>
<td>Henderson, Thomas C. and Hansen, Chuck</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $1.05 (21 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-014</td>
<td>&quot;Distributed Computation for Computer Animation.&quot;</td>
<td>Peterson, John W.</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $0.65 (13 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-015</td>
<td>&quot;Computational Fluid Dynamics for Realistic Image Synthesis.&quot;</td>
<td>Yost, Jeffrey B.</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $5.80 (116 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-016</td>
<td>&quot;CAGD-Based Computer Vision&quot;</td>
<td>Hansen, Chuck, and Henderson, Thomas C.</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $1.30 (26 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-017</td>
<td>&quot;Performance of Arc Consistency Algorithms on the CRAY.&quot;</td>
<td>Samal, Ashok, and Henderson, Thomas C.</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $1.45 (29 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-018</td>
<td>&quot;A Note on the Oslo Algorithm.&quot;</td>
<td>Lyche, Tom</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UUCS-87-019</td>
<td>&quot;Implementing Functional Programs Using Mutable Abstract Types.&quot;</td>
<td>Gopalakrishnan, Ganesh</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $1.05 (21 pages)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $0.65 (15 pages)</td>
<td></td>
</tr>
<tr>
<td>UUCS-87-021</td>
<td>&quot;A Shared Memory Algorithm and Proof for the Alternative Construct in CSP.&quot;</td>
<td>Fujimoto, Richard M. and Feng, Hwa-chung</td>
</tr>
<tr>
<td></td>
<td>Xeroxing Cost: $1.75 (35 pages)</td>
<td></td>
</tr>
</tbody>
</table>

UUCS-87-023 Fujimoto, Richard M. and Swope, Steven, "Optimal Performance of Distributed Simulation Programs." Xeroxing Cost: $.75 (15 pages)

UUCS-87-024 Gopalakrishnan, Ganesh, "A Compositional Model for Synchronous VLSI Systems." Xeroxing Cost: $1.05 (21 pages)


UUCS-87-027 Lindstrom, Gary and Bage, Goran, "Combinator Evaluation of Functional Programs with Logical Variables." Xeroxing Cost: $1.30 (26 pages)


UUCS-86-002 Boult, T. and Sikorski, K., "Complexity of Computing Topological Degree of Lipschitz Functions in N Dimensions" Xeroxing Cost: $1.15 (23 pages)

UUCS-86-003 Jacobs, Steven R. and Smith, Kent F., "PPL Quick Reference Guide (NMOS)" (Revised 1-12-87) Xeroxing Cost: $3.50 (70 pages)

UUCS-86-004 Jacobs, Steven R. and Smith, Kent F., "SIMPPL User's Guide" (Revised 1-12-87) Xeroxing Cost: $1.55 (31 pages)

UUCS-86-005 Jacobs, Steven R., Smith, Kent F., Smith, Jeff, and Smith, Kevin,"PPL Design Examples (NMOS30 Version)" (12-12-85) Xeroxing Cost: $4.65 (93 pages)

UUCS-86-006 Jacobs, Steven R. and Smith, Kent F., "Tiler User's Guide" (Revised 1-12-87) Xeroxing Cost: $1.50 (30 pages)


UUCS-86-008 Wang, Wei, Gu, Jun, and Henderson, Thomas C., "A Systolic Array of Implementation of a Discrete Relaxation Algorithm Using Path Programmable Logic." Xeroxing Cost: $2.80 (56 pages)
Rajopadhye, Sanjay V., Purushothaman, S., and Fujimoto Richard, "On Synthesizing Systolic Arrays from Recurrence Equations with Linear Dependencies." Xeroxing Cost: $0.80 (16 pages)

Carter, Tony M., "Ada to Silicon Compiler Study" Xeroxing Cost: $8.15 (163 pages)

Rajopadhye, Sanjay V. and Furimoto, Richard, "Systolic Array Synthesis by Static Analysis of Program Dependencies." Xeroxing Cost: $0.60 (12 pages)

Grodstein, Joel, "Sisyphus User's Manual." Xeroxing Cost: $1.00 (20 pages)

Hansen, Chuck and Henderson, Tom, "UTAH Range Database." Xeroxing Cost: $1.20 (24 pages)

Henderson, Thomas C. and Hansen, Chuck, "Multisensor Knowledge Systems." Xeroxing Cost: $1.05 (21 pages)

Samal, Ashok and Henderson, Tom, "Combining Symbolic and Numeric Computation of the CRAY." Xeroxing Cost: $1.05 (21 pages)


Gopalakrishnan, Ganesh C., "From Algebraic Specifications to Correct VLSI Systems." Xeroxing Cost: $13.80 (276 pages)

Grodstein, Joel. "Sisyphus - An Environment for Simulation." Xeroxing Cost: $4.90 (98 pgs)


Knell, Jeffrey Kent. "Resource Allocation In An Experimental Lisp Compiler." Xeroxing Cost: $4.75 (95 pgs)

Rajopadhye, Sanjay and Panangaden, Prakash, "Verification of Systolic Arrays: A Stream Functional Approach." Xeroxing Cost: $1.15 (23 pages)

Seppi, Larry, "Implementation of SCMAID: A Microprogrammer's Assistant for Concurrent Soft Controlled Microarchitectures." Xeroxing Cost: $7.20 (144 pages)

Mohr, Roger, and Henderson, Thomas C., "ARC and Path Consistency Revisited". Xeroxing Cost: $0.55 (11 pages)

Henderson, Tom, Hansen, Chuck, Grupen, Rod, and Bhanu, Bir, "The Synthesis of Visual Recognition Strategies" Xeroxing Cost: $0.45 (9 pages)

Sikorski, K., "Optimal Solution of Nonlinear Equations" Xeroxing Cost: $0.95 (19 pages)

Cohen, Elaine, "A New Local Basis for Designing with Tensioned Splines" Xeroxing Cost: $2.35 (47 pages)

Keller, Robert M., "Rediflow Architecture Prospectus" (date of last revision: 5 April 1986) Xeroxing Cost: $1.50 (30 pages)
UUCS-85-110  Parvin, Bahram, and Bhanu, Bir, "Segmentation of Complex Outdoor Scenes" Xeroxing Cost: $1.25 (25 pages)

UUCS-85-111  Bhanu, Bir and Ming, John C., "Recognition of 2-D Occluded Objects and their Manipulation by PUMA 560 Robot" Xeroxing Cost: $4.55 (91 pages)

UUCS-85-112  Bhanu, Bir, Ho, C.C., and Henderson, T., "3-D Model Building for Computer Vision" Xeroxing Cost: $1.05 (21 pages)

UUCS-85-113  Henderson, Tom and Samal, Ashok, "2-D Scene Analysis Using Split-Level Relaxation" Xeroxing Cost: $1.50 (30 pages)

UUCS-85-114  Bhanu, B., Lee, S.K., Ho, C.C., and Henderson, T.C, "Range Data Processing: Representation of Surfaces by Edges" Xeroxing Cost: $0.75 (15 pages)

UUCS-85-115  Tom Henderson, Chuck Hansen, Ashok Samal, C.C. Ho. and Bir Bhanu, "CAGD Based 3-D Visual Recognition" Xeroxing Cost: $0.90 (18 pages)

UUCS-85-116  Daniel S. Janik, MD, MPH, Thomas C. Henderson, PhD., John L. Lyon, MD, Ed Sharp, and August L. Jung, MD, "The Importance of Unknowns in Epidemiologic Studies" Xeroxing Cost: $0.45 (9 pages)


UUCS-85-120  Donahue, Brian Douglas. "Modeling Complex Objects With Generalized Sweeps." Xeroxing Costs: $3.75 (75 pgs)


UUCS-85-122  Kitaoka, Shoichi. "KIT: An Experimental Solid Modelling System. Xeroxing Cost: $5.35 (107 pgs)


UUCS-84-001  Shilcrat, Esther, Panangaden, Prakash, and Henderson, Thomas C., "Implementing Multi-Sensor Systems in a Functional Language". Xeroxing Cost: $0.75 (15 pages)

UUCS-84-002  Henderson, Thomas C. and Shilcrat, Esther, "Logical Sensor Systems". Xeroxing Cost: $1.60 (32 pages)

UUCS-84-003  Riesenfeld, Richard F. and Hollaar, Lee A., "Computer Aided Design." Xeroxing Cost: $1.50 (30 pages)
Keller, Robert M. and Panangaden, Prakash, "Semantics of Networks Containing Indeterminate Operators." Xeroxing Cost: $0.90 (18 pages)

Panangaden, Prakash, "A Category Theoretic Formalism for Abstract Interpretation", Xeroxing Cost: $1.30 (26 pages)

Panangaden, Prakash, Mishra, Prateek, "Abstract Interpretation and Indeterminacy" Xeroxing Cost: $0.90 (18 pages)

Pickett, Forrest B., "Simulation of Cells. Xeroxing Cost: $4.50 (90 pages)

Riesenfeld, Richard F. and Smith, Kent F., "An Experimental System for Computer Aided Geometric Design." Xeroxing Cost: $3.40 (68 pages)

Thomas, Spencer W., "Modelling Volumes Bounded by B-Spline Surfaces." Xeroxing Cost: $6.25 (125 pages)

Rajopadhye, Sanjay V., "A Formal Basis for Synthesising Systolic Arrays." Xeroxing Cost: $1.75 (35 pages)

Henderson, Thomas C. and Wu So Fai, "Pattern Recognition in a Multi-sensor Environment". Xeroxing Cost: $2.20 (44 pages)

Henderson, Thomas C. and Wu So Fai, "Some Experiments with the 3-D Hough Shape Transform". Xeroxing Cost: $1.45 (29 pages)

Henderson, Thomas C., Shilcrat, Esther, and Hansen, Charles, "A Fault Tolerant Sensor Scheme." Xeroxing Cost: $0.75 (15 pages)

Organick, Elliott I., et. al., "Transforming an Ada Program Unit to Silicon and Verifying Its Behavior in an Ada Environment: A First Experiment." Xeroxing Cost: $0.85 (17 pages)

Organick, Elliott, et. al., "Transparent Interface Between Software and Hardware Versions of Ada Compilation Units." Xeroxing Cost: $1.10 (22 pages)


Organick, Elliot I., "Transformation of Ada program Units Into Silicon." Xeroxing Cost: $1.75 (35 pages)


Lindstrom, Gary, "The Key Node Method: A Highly-Parallel Alpha-Beta Algorithm". Xeroxing Cost: $1.20 (24 pages)

Lindstrom, Gary, and Hunt, Frances C., "Consistency and Currency in Functional Databases". Xeroxing Cost: $1.25 (25 pages)

Purushothaman, S., and Subrahmanyam, P. A., "A Model for Call-By-Need and Stream Primitives in CCS". Xeroxing Cost: $0.80
<table>
<thead>
<tr>
<th>Code</th>
<th>Title</th>
<th>Author(s)</th>
<th>Xeroxing Cost:</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTEC-82-016</td>
<td>Drenan, Lawrence S., and Organick, Elliott I., &quot;ADA to Silicon Transformation: The Outline of a Method&quot;.</td>
<td>$1.80 (36 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTEC-82-020</td>
<td>Organick, Elliott I., &quot;Transformation of ADA Programs into Silicon&quot;.</td>
<td>$3.35 (67 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTEC-82-066</td>
<td>Purushothaman S., and Subrahmanyam, P. A., &quot;An Algebraic Formulation of Seitz's Weak Conditions for Self Timed Circuits&quot;.</td>
<td>$0.50 (10 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTEC-82-100</td>
<td>Keller, Robert M., and Lindstrom, Gary, &quot;Toward Function-Based Distributed Database Systems.&quot;</td>
<td>$1.90 (38 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTEC-82-101</td>
<td>Carter, Tony, &quot;ASSASSIN: A Cad System for Self-Timed Control-Unit Design&quot;.</td>
<td>$1.40 (28 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTEC-82-103</td>
<td>Organick, Elliott I., &quot;Transformation of Ada Program into Silicon.&quot;</td>
<td>$1.25 (25 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTEC-81-018</td>
<td>Baxter, Brent S., &quot;A Standard Magnetic Tape Format for Digital Image Exchange&quot;.</td>
<td>$0.75 (15 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTEC-81-020</td>
<td>Boll, Steven F., &quot;Noise Suppression Methods for Robust Speech Processing&quot;.</td>
<td>$2.50 (50 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTEC-80-004</td>
<td>Youngberg, James E., &quot;A Constant Percentage Bandwidth Transform for Acoustic Signal Processing&quot;.</td>
<td>$6.00 (120 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTEC-80-058</td>
<td>Boll, Steven F., &quot;Noise Suppression Methods for Robust Speech Processing&quot;.</td>
<td>$1.85 (37 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UUCS-80-101</td>
<td>Subrahmanyam, P. A., &quot;On Proving the Correctness of Data Type Implementation&quot;.</td>
<td>$2.20 (44 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UUCS-80-102</td>
<td>Barsky, Brian A., and Gram, Christian, &quot;A Description of Several Tools for the Synchronization of Concurrent Processes.&quot;</td>
<td>$0.95 (19 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UUCS-80-103</td>
<td>Gram, Christian, and Organick, Elliott I., &quot;Characteristics of a Functional Programming Language&quot;.</td>
<td>$2.40 (48 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UUCS-80-104</td>
<td>Barsky, Brian A., and Thomas, Spencer W., &quot;Transpline Curve Representation Systems&quot;.</td>
<td>$1.15 (23 pages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UUCS-80-105</td>
<td>Hayes, Alan B. &quot;Stored State Asynchronous Sequential Circuits&quot;</td>
<td>$0.60 (12 pages)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Weiser, Uri and Davis, Alan L., "Mathematical Representation for VLSI Arrays". Xeroxing Cost: $1.55 (31 pages)

Keller, Robert M., "Semantics and Applications of Function Graphs". Xeroxing Cost: 3.50 (70 pages)

Petersen, Tracy L., "Acoustic Signal Processing in the Context of a Perceptual Model". Xeroxing Cost: $4.85 (97 pages)

Pulsipher, Dennis C., "Application of Adaptive Noise Cancellation to Noise Reduction in Audio Signals". Xeroxing Cost: $3.50 (70 pages)


Lindstrom, Gary, "Alpha Beta Pruning on Evolving Game Trees". Xeroxing Cost: $2.60 (52 pages)


Bloomenthal, Jules I., Fish, Russell D., and Riesenfeld, Richard F., "Frame Buffer System Selection". Xeroxing Cost: $0.65 (13 pages)

Forrest, A. R., "On the Rendering of Surfaces". Xeroxing Cost: $0.95 (19 pages)

Maxey, Gregory F., and Organick, Elliott I., "CASL - A Language for Automating the Implementation of Computer Architectures". Xeroxing Cost: $1.75 (35 pages)

Boll, Steven F., and Pulsipher, Dennis C., "Suppression of Acoustic Noise in Speech Using Two Microphone Adaptive Noise Cancellation". Xeroxing Cost: $2.70 (54 pages)

Lane, J. M., and Riesenfeld, R. F., "The Application of Total Positivity to Computer Aided Curve and Surface Design". Xeroxing Cost: $4.70 (94 pages)

Lindstrom, Gary, and Soffa, Mary Lou, "Referencing and Retention in Block-Structured Coroutines". Xeroxing Cost: $2.50 (50 pages)


Boll, Steven F., "Noise Suppression Methods for Robust Speech Processing". Xeroxing Cost: $3.30 (66 pages)

Boll, Steven F., "Noise Suppression Methods for Robust Speech Processing". Xeroxing Cost: $5.50 (110 pages)


Keller, Robert M., "Sentinels: A Concept for Multiprocess Coordination". Xeroxing Cost: $1.60 (32 pages)

Davis, A. L., "Data Driven Nets: A Maximally Concurrent, Procedural, Parallel Process Representation for Distributed Control Systems". Xeroxing Cost: $2.10 (42 pages)

Boll, Steven F., "A Spectral Subtraction Algorithm for Suppression of Acoustic Noise in Speech". Xeroxing Cost: $1.50 (30 pages)

Evans, David C., "Display of Complex Three Dimensional Finite Element Models". Xeroxing Cost: $5.10 (102 pages)

Boll, Steven F., "Noise Suppression Methods for Robust Speech Processing". Xeroxing cost: $4.20 (84 pages)

Boll, Steven F., "Noise Suppression Methods for Robust Speech Processing". Xeroxing Cost: $3.90 (78 pages)

Evans, David C., "Display of Complex Three Dimensional Finite Element Models". Xeroxing Cost: $6.45 (129 pages)

Stockham, Thomas G., "Sensory Information Processing". Xeroxing Cost: $2.80 (56 pages)


Boll, Steven F., "Improving Linear Prediction Analysis of Noisy Speech by Predictive Noise Cancellation". Xeroxing Cost: $ .20 (4 pages)

Keller, Robert M., "Denotational Models for Parallel Programs with Indeterminate Operators". Xeroxing Cost: $1.40 (28 pages)

Lindstrom, Gary, "Backtracking in Generalized Control Settings". Xeroxing Cost: $1.60 (32 pages)


Boll, Steven F., "Application of the Saber Method for Improved Spectral Analysis of Noisy Speech". Xeroxing Cost: $2.00 (40 pages)

Dintelman, Sue Marie Thompson, "Design and Implementation of a Relational Data Base System for a Minicomputer" Xeroxing Cost: $4.65 (93 pages)

Keller, Robert M., "Semantics of Parallel Program Graphs". Xeroxing Cost: $1.25 (25 pages)

Smith, John J., and Smith, Diane C. P., "Integrated Specifications for Abstract Systems". Xeroxing Cost: $2.50 (50 pages)


Lindstrom, Gary, "Efficiency in Nondeterministic Control
Through Non-Forgetful Backtracking. Xeroxing Cost: $1.00 (20 pages)

| UTEC-77-135  | McDonald, Lee E., "Image Recognition Using Generalized Correlation". Xeroxing Cost: $3.75 (75 pages) |
| UTEC-76-008  | Stockham, Thomas G., "Sensory Information Processing and Symbolic Computation". Xeroxing Cost: $7.00 (140 pages) |
| UTEC-76-015  | Crow, Franklin C., "The Aliasing Problem in Computer-Synthesized Shaded Images". Xeroxing Cost: $5.20 (104 pages) |
| UTEC-76-209  | Callahan, Michael Wayne, "Acoustic Signal Processing Based on the Short-Time Spectrum". Xeroxing Cost: $4.25 (85 pages) |
| UTEC-76-226  | Christiansen, Richard W., "Word Recognition in Continuous Speech Using Linear Prediction Analysis". Xeroxing Cost: $4.30 (86 pages) |
| UTEC-76-270  | Fuchs, Henry, "The Automatic Sensing and Analysis of 3-D Surface Points from Visual Scenes". Xeroxing Cost: $4.45 (89 pages) |
| UTEC-76-271  | Atashroo, Mohammad A., "Pole-Zero Modeling and its Applications to Speech Processing". Xeroxing Cost: $5.20 (104 pages) |
| UTEC-75-047  | Parke, Frederic Ira, "A Parametric Model for Human Faces" Xeroxing Cost: $5.60 (112 pages) |
| UTEC-75-115  | Rom, Raphael Jona, "Image Transmission and Coding Based on Human Vision". Xeroxing Cost: $4.15 (83 pages) |
| UTEC-75-118  | Ingebratingsen, Robert Bergstrom, "Log Spectral Estimation for Stationary and Nonstationary Processes". Xeroxing Cost: $6.95 (139 pages) |
| UTEC-75-142  | Greer, William H., "Monaural Sensitivity to Dispersion in Impulses and Speech". Xeroxing Cost: $7.50 (150 pages) |
| UTEC-75-168  | Baxter, Brent, "Image Processing in the Human Visual System". Xeroxing Cost: $4.60 (92 pages) |
Vickers, Donald Lee, "Sorceror's Apprentice: Head-mounted Display and Wand". Xeroxing Cost: $4.25 (85 pages)

Cannon, Thomas J., "Digital Image Deblurring by Nonlinear Homomorphic Filtering". Xeroxing Cost: $4.75 (95 pages)

Clark, James Henry, "3-D Design of Free-Form B-Spline Surfaces". Xeroxing Cost: $4.05 (81 pages)

Catmull, Edwin, "A Subdivision Algorithm for Computer Display of Curved Surfaces". Xeroxing Cost: $4.10 (82 pages)

Boll, Steven F., "Selected Methods for Improving synthesis Speech Quality Using Linear Predictive Coding: System Description, Coefficient Smoothing and Streak". Xeroxing Cost: $3.15 (63 pages)

Boll, Steven F., "A Prior Digital Speech Analysis". Xeroxing Cost: $5.75 (115 pages)

Clark, Becky Jane, "A Design for a Micromodular Self-Timed System". Xeroxing Cost: $1.55 (31 pages)

Marcchal, Andre, "Data Structure Techniques in an Implementation of APL". Xeroxing Cost: $6.65 (133 pages)

Riesenfeld, Richard F., "Applications of B-Spline Approximation to Geometric Problems of computer-Aided Design". Xeroxing Cost: $4.65 (93 pages)


Wessler, Barry David, "Computer-Assisted Visual Communication". Xeroxing Cost: $5.35 (105 pages)

Debry, Roger K. and Greenfield, Harvey, "Simulation of Arbitrary Shaped Boundaries for Hemodynamic Studies" Xeroxing Cost: $14.00 (280 pages)

Phong, Bui Tuong, "Illumination for Computer Generated Images". Xeroxing Cost: $4.80 (96 pages)


Gestreicher, Donald, "Automatic Printed Circuit Board Design". Xeroxing Cost: $6.25 (125 pages)

Parke, Frederic Ira, "Computer Generated Animation of Faces". Xeroxing Cost: $2.80 (56 pages)

Archuleta, Michael, "Hidden Surface Line Drawing Algorithm". Xeroxing Cost: $3.70 (74 pages)

Burton, Robert Preece, "Real-Time Measurement of Multiple Three-Dimensional Positions". Xeroxing Cost: $6.25 (125 pages)

March, 1972

Stockham, Thomas G., "Intra-Frame Encoding for Monochrome Images by Means of a Psychophysical Model Based on Nonlinear
| UTEC-71-113 | Gouraud, Henri, "Computer Display of Curved Surfaces". Xeroxing Cost: $4.20 (84 pages) |
| UTEC-71-114 | Bennion, Scott Thomas, "A Method of Solution for Hydrodynamic and Radiation Diffusion as a Multi-Material Problem in One Dimension". Xeroxing Cost: $5.75 (115 pages) |
| UTEC-71-117 | Ashton, Alan Conway, "Electronics, Music and Computers". Xeroxing Cost: $8.85 (177 pages) |
| UTEC-70-100 | Mahl, Robert, "An Analytical Approach to Computer Systems Scheduling". Xeroxing Cost: $5.95 (119 pages) |
| UTEC-70-101 | Watkins, Gary Scott, "A Real Time Visible Surface Algorithm" Xeroxing Cost: $4.75 (95 pages) |
| UTEC-70-103 | Withrow, Carol, "A Dynamic Model for Computer-Aided Choreography". Xeroxing Cost: $3.00 (60 pages) |
| UTEC-70-104 | Newman, William M., "An Experimental Display Programming Language for the PDP - 10 Computer". Xeroxing Cost: $2.35 (47 pages) |
| UTEC-70-106 | Nelson, Stanley Logan, "Master Schedule Building and the Flexibility Scheduled School". Xeroxing Cost: $3.80 (76 pages) |
| UTEC-70-108 | Seror, Dennis E., "DCPL, A Distributed Control Programming Language". Xeroxing Cost: $9.15 (183 pages) |

1968
TR 4-5
Warnock, John E., "A Hidden Line Algorithm for Halftone Picture Representation". Xeroxing Cost: $0.70