Formal Specification

A Very First Acquaintance

© 2003, 2004 - Dr. Ernest Cachia
What is Formal Specification

- The application of set theory, propositional and predicate calculi to the specification of systems.
Sets

- Are collections of elements
- Are represented by standard notation
- Are manipulated by standard elementary operations
- Are entities which can interact with each other
Examples of Sets

Notation: \{ \}

- Set of colours: \{green, blue, yellow\}
- Set of sports: \{tennis, football, equestrian\}
- Set of lecturers in this room: \{ernest\}
- Empty set: \{\} or \(\emptyset\)
Set Construction

- Direct (as in previous slide)
- Operations on other sets according to the form:
  The resulting set whose elements are formed by the operation on elements selected by the condition from the original set with elements from range.

Translates to…

\[ \{ \text{set} : \text{range} \mid \text{condition} \cdot \text{operation} \} \]

- In notational form (aka comprehensive specification):

\[ \{ \text{Signature} \mid \text{Predicate} \bullet \text{Term} \} \]

\[ \{ x : X \mid P(x) \cdot E(x) \} \]
Set Construction Examples

- Alternate even numbers:
  \[ \{ x : \mathbb{N} \mid x \pmod{2} = 0 \cdot 2 \times x \} = \{0,4,8,12,16,20,\ldots\} \]

- Tens:
  \[ \{ x : \mathbb{N} \mid x \cdot 10 \times x \} = \{0,10,20,30,40,\ldots\} \]

- Squares of multiples of 4 (excluding zero):
  \[ \{ x : \mathbb{Z} \mid (x \pmod{4} = 0) \land (x > 0) \cdot x \times x \} \]
  \[ = \{16,64,144,256,\ldots\} \]
Some Set-Based Specification Exercises

Write 3 elements from the sets specified by the following comprehensive specifications:

- \( \{ n : N \mid n < 20 \land n > 10 \cdot n \} \)
- \( \{ n : N \mid n^3 > 10 \cdot n \} \)
- \( \{ x, y : N \mid x + y = 100 \cdot (x, y) \} \)
- \( \{ x, y : N \mid x + y = 5 \cdot x^2 + y^2 \} \)

Interpret the following:

- \( \{ m : monitors \mid \text{MonitorState}(m, on) \cdot m \} \)
- \( \{ f : \text{SysFiles} \mid f \in \text{DelFiles} \land f \in \text{ArcFiles} \cdot f \} \)

Write the comprehensive specification of:

\( \{(10,100),(11,121),(12,144),(13,169),(14,196)\} \)
Set Operators

- If more explanation than what was given in class is required, operator definitions can be found in any basic math textbook or Internet sources.

- Examples:
  - Mosta $\in \{\text{Maltese towns}\}$
  - London $\notin \{\text{Maltese towns}\}$
  - #$\{\text{joe, veronica, mark}\} = 3$
  - $\{\text{paul, richard, claire, george}\} \subseteq \{\text{Group B}\}$
  - $\text{IP}\{\text{m}ilan,\text{juve}\} = \{\{\},\{\text{m}ilan\},\{\text{juve}\},\{\text{m}ilan,\text{juve}\}\}$
  - $\{\text{dobie, collie, poodle}\} \cup \{\text{poodle, labrador}\} = \{\text{dobie, collie, poodle, labrador}\}$
  - $\{\text{dobie, collie, poodle}\} \cap \{\text{poodle, labrador}\} = \{\text{poodle}\}$
  - $\{\text{dobie, collie, poodle}\} \setminus \{\text{poodle, labrador}\} = \{\text{dobie, collie}\}$
Propositional Calculus

- Deals with logic
- Basically consists of statements which can be true or false (Excluded Middle Law)
- Are mutually exclusive - never true or false at the same time (Contradiction Law)
- Is fundamental
- Is axiomatic
- In theory, can be used to describe anything
Proposition (and not) Examples

- Some birds can fly
- The nation Malta is in Asia
- Dogs are mammals
- All fish live in water
- All fish live in sea water
- Mary is the only lady in our group
- Cikku qatt ma jgorr

- Sit down.
- How are you today?
- Get my tea, please.
- What is the weather like?
Representing Propositions

Consider the following…

- 10 is greater than 8
- 8 is less than 10
- $8 < 10$
- $10 > 8$
- There is a positive number such that if we add it to eight the result would be ten.

All the above are one and the same proposition.
Propositional Calculus Notation

Please refer to the R. Pressman textbook, or any other Software Engineering textbook containing basic formal specification, for a listing of basic propositional operators.
Examples to Discuss

\[ \neg((P \lor Q) \Rightarrow Q) \]
\[ \neg((P \land Q) \lor \neg R) \Leftrightarrow P \]
\[ \neg P \land (P \lor (Q \Rightarrow P)) \]
\[ ((P \Rightarrow Q) \land (R \Rightarrow S) \land (P \lor R)) \Rightarrow (Q \lor S) \]

Taking:
P as true
Q as false
R as false
S as true

Click again for answers… **Answers:**
1. TRUE
2. FALSE
3. FALSE
4. FALSE
Contradictions and Tautologies

- A *contradiction* is a proposition that is always false for all possible values and variables making it up.
- A *tautology* is a proposition that is always true for all possible values and variables making it up.
- Examples:
  
  \[ a \land \neg a \]  
  contradiction

  \[ a \lor \neg a \]  
  tautology

  \[ (a \land b \land c) \Rightarrow (c \Rightarrow a) \]  
  tautology
De Morgan’s Laws

\[(\neg(P \land Q) \iff \neg P \lor \neg Q)\]

\[(\neg(P \lor Q) \iff \neg P \land \neg Q)\]

The above can be summarised (in Maltese) as follows:


…g]a[nuha ftit f’mo]]kom!
Specifying with Propositions

Consider the following text:
The system is in alert state only when it is waiting for an intruder and is on practice alert.
\[ \text{alert} \iff \text{waiting} \land \text{practice} \]
If the system is in teaching mode and is on practice alert, then it is in alert state.
\[ \text{teaching} \land \text{practice} \Rightarrow \text{alert} \]
The system will be in teaching mode and on practice alert and not waiting for an intruder.
\[ \text{teaching} \land \text{practice} \land \neg \text{alert} \]

Therefore…
Detecting Contradictions

From the previous system:
alert $\iff$ waiting $\land$ practice
teaching $\land$ practice $\implies$ alert
teaching $\land$ practice $\land$ $\neg$alert

The second and third propositions yield a contradiction:
teaching $\land$ practice $\implies$ alert
teaching $\land$ practice $\land$ $\neg$alert
alert $\land$ $\neg$alert …contradiction!
The first proposition yields another contradiction:
teaching $\land$ practice $\land$ $\neg$(waiting $\land$ practice) eventually simplifies to…
alert $\land$ $\neg$waiting …contradiction because alert requires waiting!

© 2003, 2004 - Dr. Ernest Cachia
Predicate Calculus

- Can be viewed as conditional statements obeying propositional behaviour with specific values.
- Consider a triangle…
- We can say the following…
  1. Any side will be greater than zero length;
  2. The sum of the length of any two sides will be greater than the length of the remaining side. Therefore…
In predicate calculus form the basic properties of a triangle could be written as follows:

\[
\begin{align*}
\text{Property 1} & : a > 0 \land b > 0 \land c > 0 \land a + b > c \land b + c > a \land a + c > b \\
\text{Property 2} & :
\end{align*}
\]
Consider This Statement

Fido is a dog, dogs like bones so Fido likes bones.

Propositional analysis of this sentence yields:
Fido is a dog     (propos. 1) P  
Dogs like bones  (propos. 2) Q
Fido likes bones (propos. 3) R

Can we derive R from P and Q using only propositional calculus? – No!

Therefore...
We Introduce “A Predicate”

- Formally a predicate can be seen as direct indicators of object properties.
- Examples of unary predicates:
  - dog(fido) = true
  - dog(lecturer) = false
- Examples of n-ary predicates:
  - father(john, mary)
  - team(pawlu, bertu, ensu)
A More Familiar Predicate Form

Consider the predicates:

- numerically_bigger_than(x, y)
- are_equal(a, b)

Can be written as…

- $x > y$
- $a = b$
Quantification

- Places bounds on free variables (i.e. names of objects)

\[ P(x) \quad \text{Is a unary predicate} \]

\[ \exists x \ gP(x) \quad \forall x \ gP(x) \quad \text{Are propositions} \]

1. There exists an object ‘x’ to which the predicate ‘P(x)’ applies.
2. For all objects ‘x’, the predicate ‘P(x)’ applies.
Quantification Examples

\[ \exists i : 1..10 \; g_i^2 = 64 \]
\[ \exists \text{proc} : \text{processors} \; g\text{ProcessorState}(\text{proc}, \text{active}) \]
\[ \exists i : \exists \; g_i > 10 \land \text{MonitorTemp} = i \]
\[ \exists m : \text{AllocatedMonitors} \; g\text{MonState}(m, \text{ready}) \]
\[ \exists i : 1..100; \exists m : \text{AllocatedMonitors} \; g\text{activity}(m, \text{functioning}) \land \text{AmbientTemp} = i \]
\[ \exists r : \text{CurrentReactors}; \exists m : \text{AllocatedMonitors} \; g\text{MonState}(m, \text{functioning}) \land \text{connected}(r, m) \]
Say the Following in Natural Language (loosely adopted from Behforooz, A.)

\[ \forall x, y, z \ g x > y \land y > z \Rightarrow x > z \]
\[ \exists x \ \exists x > 10 \lor x + y < 100 \]
\[ \forall x, y \in N \rightarrow x + y \in N \]
\[ \exists x, y \in \{1, 2, 3, 4\} \ \exists x + y \in \{1, 2, 3, 4\} \]
\[ \forall x, y \in \{1, 2, 3, 4\} g x > y \Rightarrow x - y \in \{1, 2, 3, 4\} \]
\[ \sim (p \land q) \iff \sim p \lor \sim q \]
\[ x > y \iff x - y > 0 \]
\[ x + y > 0 \nRightarrow x > 0 \land y > 0 \]
Algebraic Specifications

- A specification technique mainly used for abstract data types
- Based on a strong mathematical foundation – namely algebra
- Have been in use for relatively long periods of time
- Are universal in their application
- Employ fundamental principles
Building Algebraic Specifications

- Clearly comprehend the system to model
- Determine the operations necessary for the system you have in mind
- Specify the relationship between the system’s operations
- Write the specification down according to adopted standard (a popular standard is the Common Algebraic Specification Language – CASL)
Algebraic Specification Queue Example (1/2) (loosely adopted from Pressman, R. S.)

A message queue:

Operations:
- add an item [AddItem]
- remove an item [RemItem]
- check if empty [IsEmpty]
- get first queue item [GetFirst]
- get last item [GetLast]
- create a new queue [Create]
Algebraic Specification Example

(2/2) (loosely adopted from Pressman, R. S.)

<table>
<thead>
<tr>
<th>Type:</th>
<th>queue(Z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imports:</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signatures:</th>
<th>Axioms:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create → queue(Z)</td>
<td>isEmpty(Create)=true</td>
</tr>
<tr>
<td>AddItem(Z,queue(Z)) → queue(Z)</td>
<td>isEmpty(AddItem(z,q))=false</td>
</tr>
<tr>
<td>RemItem(Z,queue(Z)) → queue(Z)</td>
<td>RemItem(AddItem(z,q),q)=q</td>
</tr>
<tr>
<td>GetFirst(queue(Z)) → Z</td>
<td>GetLast(AddItem(z,q))=z</td>
</tr>
<tr>
<td>GetLast(queue(Z)) → Z</td>
<td>GetFirst(AddItem(z,Create))=z</td>
</tr>
<tr>
<td>isEmpty(queue(Z)) → Boolean</td>
<td></td>
</tr>
</tbody>
</table>
A symbol table:
Operations: create a new empty table [Create]
  add an item [AddItem]
  remove an item [RemItem]
  check if a symbol is in a table [InTab]
  join two tables [Join]
  get common symbols [Common]
  check if two tables are identical [isEqual]
  check if a table is part of another [IsPartof]
  check is a table is empty [IsEmpty]
Algebraic Specification Table Example (2/4) (loosely adopted from Pressman, R. S.)

- Signatures:
  Create → table(Z)
  AddItem(Z,table(Z)) → table(Z)
  RemItem(Z,table(Z)) → table(Z)
  InTab(Z,table(Z)) → Boolean
  IsPartOf(table(Z),table(Z)) → Boolean
  IsEqual(table(Z),table(Z)) → Boolean
  IsEmpty(table(Z)) → Boolean
  Join(table(Z),table(Z)) → table(Z)
  Common(table(Z),table(Z)) → table(Z)
Algebraic Specification Table Example (3/4) (loosely adopted from Pressman, R. S.)

Some applicable axioms (1):
AddItem(s2,AddItem(s1,s))=if s1=s2 then AddItem(s1,s) else AddItem(s1,AddItem(s2,s))
RemItem(s1,Create)=Create
RemItem(s1,AddItem(s2,s))=if s1=s2 then RemItem(s1,s) else AddItem(s2,RemItem(s1,s))
InTab(s1,Create)=false
InTab(s1,AddItem(s2,s))=if s1=s2 then true else InTab(s1,s)
Join(s,Create)=s
Join(s, AddItem(s1,t))=AddItem(s1,Join(s,t))
Some applicable axioms (2):
Common(s, Create) = Create
Common(s, AddItem(s1, t)) = if InTab(s1, s) then
  AddItem(s1, common(s, t)) else common(s, t)
IsEmpty(Create) = true
IsEmpty(AddItem(s1, t)) = false

IsPartOf(Create, s) = true
IsPartOf(AddItems(s1, s), t) = if InTab(s1, t) then
  IsPartOf(s, t) else false
IsEqual(s, t) = IsPartOf(s, t) \& IsPartOf(t, s)

© 2003, 2004 - Dr. Ernest Cachia
“Club” example will be discussed during lectures.
The Z-Specification Language

• Attempts to place a notational framework on formal system specification
• Based on set theory
• Is model-based (relies on well understood mathematical entities and their relationship)
• Equally used to model (specify) state as well as operations on states
Some Basic Z-Schema Examples (1/2)

\[
a : \mathbb{N}
b : \{7, 1, 3, 24\}
\]

\[
a \in b
\]

\[
a \text{ is a natural number and } b \text{ is a set formed of natural numbers as shown. } a \text{ is contained in } b.
\]

Note: Generically, to indicate “a set of”, the notation “P (with a hollow stem)”. E.g. \( b: P \mathbb{N} \)
Linear equivalent would be: \([a: \mathbb{N}; b: \{7, 1, 3, 24\} / a \in b]\)
Some Basic Z-Schema Examples (2/2)

\[ a, b : Y \]
\[ c : P \]
\[ a \in c \]
\[ b \in c \]

Is equivalent to…

\[ a, b : Y \]
\[ c : P \]
\[ a \in c \land b \in c \]

Or linearly… \[ a, b : Y ; c : P \ Y \mid a \in c \land b \in c \]
Naming Schemas

\[ MonCondition \]

\[ MonNo : ¥ \]

\[ AvailableMonitors : P ¥ \]

\[ MonNo \in AvailableMonitors \]

Or...

\[ MonCondition A \]

\[ [MonNo : ¥ ; AvailableMonitors : P ¥ | MonNo \in AvailableMonitors] \]
Please refer to R. Pressman or other mainstream Software Engineering textbook.
Z-Schema Conventions

- Delta
  - Denoted by the Greek literal (Δ)
  - Used to extend the schema components to indicate update operations, i.e. changes in state variables (updating operations).

- “Xi”
  - Denoted by the Greek literal (Ξ)
  - Used to indicate that stored data is not affected, i.e. enquiry operations.
Specify a system which will keep track of students who have handed in Assignments. There are clearly three sets involved...

\textbf{Class} \hspace{1cm} (all the students in the class)

\textbf{HandedIn} \hspace{1cm} (all the students in the class who have handed in their assignment)

\textbf{NotHandedIn} \hspace{1cm} (all the students in the class who have not handed in their assignment)

\[
\Delta \text{Assignment} \hspace{1cm}
\]

\begin{align*}
\text{Class, HandedIn, NotHandedIn} & : P \text{ STUDENTS} \\
\text{Class}', HandedIn', NotHandedIn' & : P \text{ STUDENTS} \\
\text{HandedIn} \cup \text{NotHandedIn} & = \text{Class} \\
\text{HandedIn} \cap \text{NotHandedIn} & = \emptyset \\
\text{HandedIn}' \cup \text{NotHandedIn}' & = \text{Class}' \\
\text{HandedIn}' \cap \text{NotHandedIn}' & = \emptyset
\end{align*}
Use of a Delta Schema (based on previous example)

Model the handing in of a student assignment:

\[
\begin{align*}
\text{HandIn} & \\
\text{stud} & \in \text{STUDENTS} \\
\Delta \text{Assignment} & \\
\text{Stud} & \in \text{NotHandedIn} \\
\text{NotHandedIn}' & = \text{NotHandedIn}\setminus\{\text{Stud}\} \\
\text{HandedIn}' & = \text{HandedIn} \cup \{\text{Stud}\} \\
\text{Class}' & = \text{Class}
\end{align*}
\]
Use of a “Xi” Schema (based on previous example)

Model a query for the number of students who have handed in:

\[ \exists \text{ Assignment} \]
\[ \text{Class, HandedIn, NotHandedIn : P STUDENTS} \]
\[ \text{Class', HandedIn', NotHandedIn': P STUDENTS} \]

NotHandedIn' = NotHandedIn
HandedIn' = HandedIn
Class' = Class

Therefore:

AssignQuery A

[HandedIn!:¥ ; \( \exists \) Assignment\|HandedIn! = # HandedIn]
A simple example of this will be presented during lectures.
Another Schema Inclusion Example (1/2)

FileStatus

AllFiles, FreeFile, FilesInUse : \( P \) FILES
File : FILES
RegisteredUsers : \( P \) NAMES
User : NAMES

User \( \in \) RegisteredUsers
File \( \in \) AllFiles
AllFiles = FreeFiles \( \cup \) FilesInUse
FreeFile = AllFiles \( \setminus \) FilesInUse
FreeFiles \( \cap \) FilesInUse = \( \emptyset \)
FreeFile \( \not\in \) FilesInUse

UserStatus

FileStatus
InvalidUsers : \( P \) NAMES

User \( \in \) RegisteredUsers
User \( \not\in \) InvalidUsers
RegisteredUsers \( \cap \) InvalidUsers = \( \emptyset \)
Another Schema Inclusion Example

Results in the following schema...

\[ \text{FileAndUserStatus} \]

\[ \begin{align*}
\text{AllFiles}, \text{FreeFile}, \text{FilesInUse} : & \ P \ \text{FILES} \\
\text{File} : & \ P \ \text{FILES} \\
\text{RegisteredUser}, \text{InvalidUser} : & \ P \ \text{NAMES} \\
\text{User} : & \ P \ \text{NAMES} \\
\text{User} & \in \ \text{RegisteredUsers} \\
\text{User} & \notin \ \text{InvalidUsers} \\
\text{RegisteredUsers} & \cap \text{InvalidUsers} = \emptyset \\
\text{File} & \in \ \text{AllFiles} \\
\text{AllFiles} & = \text{FreeFiles} \cup \text{FilesInUse} \\
\text{FreeFile} & = \text{AllFiles} \setminus \text{FilesInUse} \\
\text{FreeFiles} & \cap \text{FilesInUse} = \emptyset \\
\text{FreeFile} & \notin \text{FilesInUse}
\end{align*} \]
Another Schema Inclusion Example \textit{(taken from Ince)}

\begin{align*}
\text{SetInv} \\
upper, lower : P \, ¥ \\
\text{MaxSize} : ¥ \\
\#upper + \#lower \leq \text{MaxSize}
\end{align*}

\begin{align*}
\text{MidInv} \\
\text{MidInv} \\
middle : P \, ¥ \\
\text{SetInv} \\
middle \subseteq \text{upper} \cup \text{lower}
\end{align*}

The above schemas result in…

\begin{align*}
\text{MidInv} \\
\text{MidInv} \\
middle : P \, ¥ \\
\text{upper}, \text{lower} : P \, ¥ \\
\text{MaxSize} : ¥ \\
middle \subseteq \text{upper} \cup \text{lower} \\
\#upper + \#lower \leq \text{MaxSize}
\end{align*}
Practical Z-Spec Example (elevator) *(taken from Ghezzi)*

This example is produced as a separate MS-Word document offered as a separate hand-out which should be circulated with this material.
Sequences

- Are not sets
- Can be viewed as collections with predefined constraints
- Exist in different forms
- Can have operations applied to them
- Widespread use in computer systems
Types of Sequences

- Normal (including empty) $seq$
- Non-empty $seq_1$
- Injective (not containing duplicates) $iseq$

Therefore…
Formal Sequence Definitions

- Definitions...
  \[ \text{seq } T = \{ f : \mathbb{N} \rightarrow T \mid \text{dom } f = 1 \ldots \#f \} \]
  \[ \text{seq}_1 T = \{ f : \text{seq } T \mid \#f > 0 \} \]
  \[ \text{iseq } T = \text{seq } T \cap (\mathbb{N} \rightarrow T) \]

- Some examples...
  E.g. of (seq N) is \{1\alpha3, 2\alpha9, 3\alpha9, 4\alpha11\}
  written as \langle3,9,9,11\rangle
  E.g. of (iseq files) is \{1\alpha UpdateFile, 2\alpha LogFile, 3\alpha TaxFile\}
  written as \langle UpdateFile, LogFile, TaxFile\rangle
Sequences in $\mathbb{Z}$

Examples of these will be presented during lectures.
Sequence (in Z) Example

---

**FileQueue**

- **InQueue, OutQueue : seq Files**
- \(#\text{InQueue} < #\text{OutQueue}\)

---

**Rentals**

- **Pending, Overdue : seq ID**
- **MostOverdue!, SoonToBeOverdue! : ID**

**MostOverdue! = head Overdue**
- **SoonToBeOverdue! = head Pending**
Can you try to formally define the “head”, “last”, “tail”, and “front” sequence operators using a Z-schema?

\[
\begin{align*}
\text{SeqOps} & \\
\text{head, last} : & \text{seq}_1 \text{SeqOps} \to \text{SeqOps} \\
\text{tail, front} : & \text{seq}_1 \text{SeqOps} \to \text{seq SeqOps}
\end{align*}
\]

\[
\forall s : \text{seq}_1 \text{SeqOps} \bullet \\
\quad \text{head } s = s(1) \land \\
\quad \text{last } s = s(\#s) \land \\
\quad \text{tail } s = (\lambda n : 1 \ .. \ \#s-1 \bullet s(n+1)) \land \\
\quad \text{front } s = (1 \ .. \ \#s-1) \triangleleft s
\]