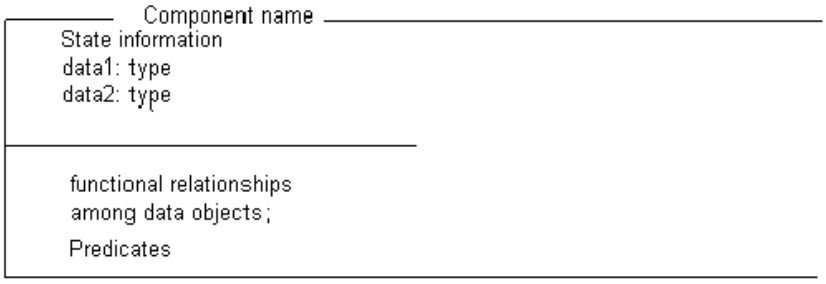
## Specification language Z

Defines the data, types, functions, and relationships among these in a specification
General syntax:






## Some notation

ΞX – State X doesn't change

ΔX – State X changes

ΘS'=ΘS – S invariant

ℙ X – a set of X; 𝔽 X – a finite set of X

dom – domain; ran – range

f: X –+–> Y – a function from X to Y

f: X >–+–> Y – a partial injection (Not every X is defined and each Y has a unique X)

f ⊕ {X |––> Y} – f appended with f(X) = Y

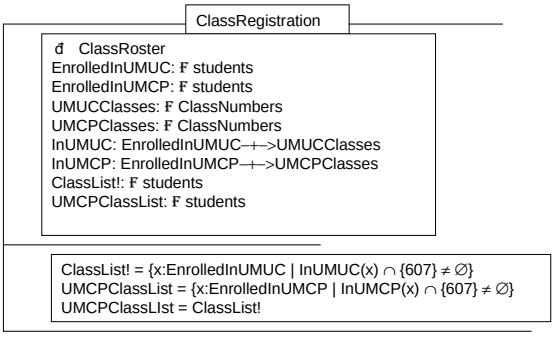
{X} ◁ f – f with x removed from domain of f

x' – final value of x

x? – x is an input data item

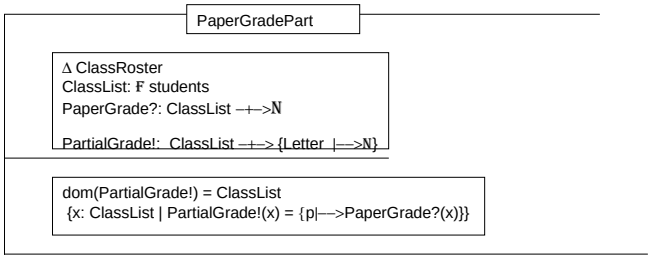
x! – x is an output data item

# Z Example – MSWE 607 grading (1998 class)



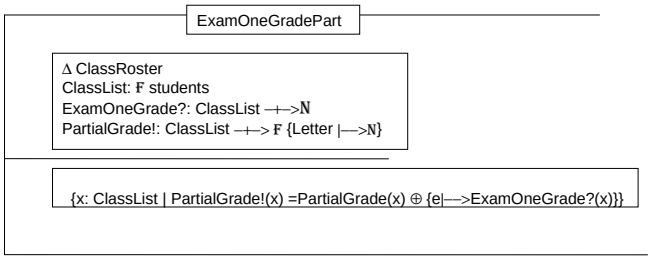

  

# Z specification - 2



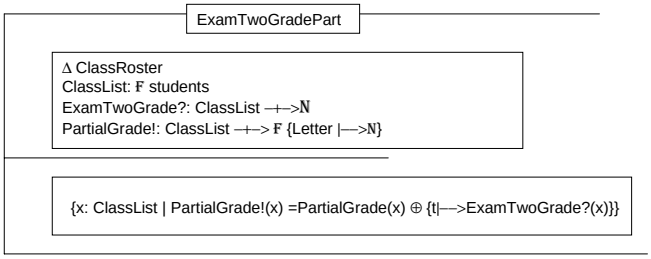

  

# Z specification - 3



# Z specification - 4



# Z specification - 5

ExamTwoGradePart

Δ ClassRoster
ClassList: $\mathbb{F}$ students
ExamTwoGrade?: ClassList $\rightarrow\!\!\!\!\!+\!\!\!\rightarrow \mathbb{N}$
PartialGrade!: ClassList $\rightarrow\!\!\!\!\!+\!\!\!\rightarrow \mathbb{F}$ {Letter $\mid\!\!\rightarrow \mathbb{N}$}

{x: ClassList | PartialGrade!(x) =PartialGrade(x) ⊕ {t|—>ExamTwoGrade?(x))}



# Z specification - 6

FinalGradePart

ClassRoster
ClassList: $\mathbb{F}$ students
Curve: $\mathbb{R}$ $\rightarrow\!\!\!\!\!+\!\!\!\rightarrow$ Letter
PartialGrade: ClassList $\rightarrow\!\!\!\!\!+\!\!\!\rightarrow \mathbb{F}$ {Letter $\mid\!\!\rightarrow \mathbb{N}$}
FinalGrade!: ClassList $\rightarrow\!\!\!\!\!+\!\!\!\rightarrow$ Letter

dom(FinalGrade!) = ClassList
ran(FinalGrade!) = {A,B,C,F}
{x:ClassList | dom(PartialGrade(x))={p,e,t,f}
Curve(X) = (X≥90⇒A) ∪ ( X≥80∩X<90⇒B) ∪ ( X≥70∩X<80⇒C) ∪ (X<70⇒F)
{x: ClassList | FinalGrade!(x) = Curve(.10*PartialGrade(x)(p) +.25*PartialGrade(x)(e)
+.25*PartialGrade(x)(t) +.40*PartialGrade(x)(f))

## Homework !

- Fix up previous Z specification to reflect grading this semester in MSWE 607.

## Practical Examples of Formal Methods

Use of state tables to define requirements:

**Example – From NASA – JPL: 2.16.3.f) While acting as the bus controller, the C&C MDM CSCI shall set the e, c, w, indicator identified in Table 3.2.16-II for the corresponding RT to "failed" and set the failure status to "failed" for all RT's on the bus upon detection of transaction errors of selected messages to RTs whose 1553 FDIR is not inhibited in two consecutive processing frames within 100 msec of detection of the second transaction error if; a backup BC is available, the BC has been switched in the last 20 sec, the SPD card reset capability is inhibited, or the SPD card has been reset in the last 10 major (10 sec) frames, and either:**

**1. the transaction errors are from multiple RTs, the current channel has been reset with the last major frame, or**

**2. the transaction errors are from multiple RT's, the bus channel's reset capability is inhibited, and the current channel has not been reset with the last major frame.**

## State table solution to above requirements: When to switch RTs.

**And—Or table. Read conditions down ("and" them). Each column defines a separate ("or) sequence of actions. (● – don't cares)**

| | | | | |
|---|---|---|---|---|
| **C&C MDM acting as bus controller** | **T** | **T** | **T** | **T** |
| **Detection of transaction errors in 2 consecutive processing frames** | **T** | **T** | **T** | **T** |
| **Errors are on selected messages** | **T** | **T** | **T** | **T** |
| **The RTs 1553 FDIR is not inhibited** | **T** | **T** | **T** | **T** |
| **A backup BC is available** | **T** | **T** | **T** | **T** |
| **The BC has been switched in the last 20 sec** | **T** | **T** | **T** | **T** |
| **SPD card reset capability is inhibited** | **T** | **T** | ● | ● |
| **SPD card has been reset in the last 10 major (10 sec) frames** | ● | ● | **T** | **T** |
| **Transaction errors are from multiple RTs** | **T** | **T** | **T** | **T** |
| **The current channel has been reset within the last major frame** | **T** | **F** | **T** | **F** |
| **Bus channel's reset capability is inhibited** | ● | **T** | ● | **T** |



## PVS Example: NASA Cassini spacecraft

Requirement: "If spacecraft safing is requested via a CDS (command and data subsystem) internal request while the spacecraft is in a critical attitude, then no change is commanded to the AACS (attitude and articulation control system) attitude. Otherwise the AACS is commanded to the homebase attitude."

AacsStopFnc requirement and "requirements met" Lemma

## Cassini proof

```
Saf: THEORY
BEGIN
AacsMode: TYPE = {homebase, detumble}
Attitude:       TYPE
CdsIntReq: VAR bool
CritAtt: VAR bool
PrevAacsMode: VAR MacsMode
AacsStopFnc(CritAtt, CdsIntReq, PrevAacsMode):
AacsMode =     IF CritAtt
                              THEN IF CdsIntReq
                                      THEN PrevAacsMode
                                      ELSE homebase
                                      ENDIF
                              ELSE homebase
                              ENDIF
AacsSafingReqMet1: LEMMA
  (CritAtt AND CdsIntReq) OR
    (AacsStopFnc(CritAtt,CdsIntReq,PrevAacsMode)
    = homebase)
END saf
```



## A Case study in Specification Failure – Ariane 5

**Facts:**

**Design issues: In order to save funds and ensure reliability, and since the SRI is used for prelaunch activities, and since the French Ariane 4 was a successful rocket, the SRI from Ariane 4 was reused for the Ariane 5.**

**Operations: On June 4, 1996 the Ariane 5 launch vehicle failed 39 seconds after liftoff causing the destruction of over $100 million in satellites**

**Cause of failure: The Inertial Reference System (SRI), which controls the attitude of the vehicle by sending aiming commands to the rocket nozzle, sent a bad command to the rocket causing the nozzle to move the rocket toward the horizontal.**

**The vehicle tried to switch to the backup SRI (computer), but that failed for the same reason 72 millisec earlier.**

**The vehicle had to then be destroyed.**

## Inquiry Board Findings

- **SRI tried to convert a floating point number out of range to integer. Therefore it issued an error message (as a 16 bit number). This 16 bit number was interpreted as an integer by the guidance system and caused the nozzle to move accordingly.**
- **The backup SRI performed according to specifications and failed for the same reason.**
- **The Ada range checking was disabled since the SRI was supposedly processing at 80% load and the extra time needed for range checking was deemed unnecessary since the Ariane 4 software worked well.**
- **The ultimate cause of the problem was that the Ariane 5 has a more pronounced angle of attack and can move horizontally sooner after launch. The "bad value" was actually the horizontal speed of the vehicle.**
- **The SRI was used for prelaunch activities. It could have been turned off as soon as launch began.**



## Was this a software failure?

**No failure mode on SRI. Besides "crash only," needed "omission" and "non malicious behavior" modes (state tables?)**

**No range checking on data. No validation that OBC gets correct data in face of SRI failure (verification?).**

**System engineering specifications – guidance on Ariane 5 different from Ariane 4. Reuse is not free. Just because Ariane 4 SRI was bug free, reuse of code is not relevant if it is not the right product.**

**Pair of SRI systems. Assumed only random hardware errors – no account for any software problems. 1 out of 2 not sufficient.**

Inquiry Board claimed this was a software engineering failure, but others claimed it was a systems engineering failure. Software Engineers did all they were supposed to be able to do.

Comments?