Chapter 7

An Example: Playing Golf

It has been suggested by colleagues [Mel00, Lev00] that safety concepts can be applied to a simple system such as a golf game. Recall that all that is needed to have a system is objects and behavior.

Intuitive Interpretation The idea is that

- an accident is some event such as playing too many strokes (and thereby losing the game);
- Landing in a bunker entails with high probability that one increases the number of strokes required to play the hole, and thus to play the game
- Thus landing in a bunker, or being in a bunker, represents a hazard
- The risk associated with the hazard (with the bunker) is the expected number of extra strokes one must take to get out of the bunker

7.1 The Basics: Objects, Predicates, Accident

We provide here a reconstruction of the example according to the concepts we have already introduced.

Objects The system and environment are defined by objects. These are

- A golf ball: b
- A player: p
- A course: C
- A bunker: B

Let us for simplicity assume that there is only one bunker. Notice that the bunker is *part of* the course: $B \prec C$.

Predications There really isn't a whole lot that can be said yet in the way of predications. Intuitively,

- the ball can be on the course: $loc(b, C)$
- and when on the course it can be in a bunker: $loc(b, B)$
- the player can be on the course: $loc(p, C)$
- and the player can be in the bunker: $loc(p, B)$

Accident An accident is too high a stroke count at the end. It seems we need a fluent *TotalStrokes* for the total number of strokes played. We have simplified by having only one player, so the player must be playing against a set total, N. An accident would be a total stroke count greater than this limit: $TotalStrokes > N$.

7.2 The System And Behavior

We only have four main objects, so there are only a few choices.

- the ball b alone belongs to the system; the other objects to the environment. This would mean that the predications above are all hybrid or environment predicates:
  - $loc(b, C)$ and $loc(b, B)$ are hybrid;
  - $loc(p, C)$ and $loc(p, B)$ are environment.

  System predicates would be those obtained from the hybrid predicates by quantification. There is only one, namely
  - *Exists* $X.loc(b, X)$

  And if we assume that the ball remains on the course the entire time, this sole system predicate turns out to be always true.
- The ball b and player p belong to the system. It follows that all four predicates above are hybrid, and the system predicates are *Exists* $X.loc(b, X)$ and *Exists* $X.loc(p, X)$ If we assume that the player remains on the course with the ball, both of these are always true.
- The course C belongs to the system, player and ball to the environment. Because B is part of C, $B \prec C$, B must belong to the system also. The system predicates would be *Exists* $x.loc(x, C)$ and *Exists* $x.loc(x, B)$: there is a player or ball on the course or in the bunker. Again, the former is always true; the latter is what we intuitively have called a hazard.

- The bunker B belongs to the system; the course C to the environment. Whether ball and player belong to system or environment, one may classify the predicates similarly to above.
- Everything belongs to the system. In this case, all predicates are system predicates.

Additional Objects In order to say what we meant by an accident, we introduced the fluent *TotalStrokes* and the natural numbers (or at least one of them, N). If we include the natural numbers, we have more to say. In principle, we should inquire whether the fluent *TotalStrokes* and the natural numbers belong to the system or not. In practice, it doesn't matter. In general, expressive artifacts we introduce in order to be able to talk about a system will not need to be classified with the system, but in certain circumstances they may.

Greater Expressive Capability Accumulating strokes by landing in a bunker raises the chances of *TotalStrokes* exceeding the target N, because of the increased chance of extra strokes being needed. But one extra stroke in a bunker may be neutralised by a reduced number of strokes (due to luck or skill) later. So it's not inevitable that an accident will occur if one lands in a bunker.

Behavior We must say what kind of behavior the system can engage in (whatever we take the system to be). In order to specify behavior, we have to say what changes can occur, of system and environment. The ball and player are always on the course, so there can be no change in this predication. However, both ball and player can be in or out of the bunker. This gives the possibility of four changes. In the formalism below, I use the prime symbol " $'$ " on a predicate to indicate that this is true *after* the change, and any predicate without a prime is asserted to be true *before* the change.

- Player in bunker, then player out:

$$loc(p, B) \;\&\; \neg loc'(p, B)$$

- Player out of bunker, then in:

$$\neg loc(p, B) \;\&\; loc'(p, B)$$

- Ball in bunker, then out:

$$loc(b, B) \;\&\; \neg loc'(b, B)$$

- Ball out of bunker, then in:

$$\neg loc(b, B) \;\&\; loc'(b, B)$$

Furthermore, strokes are being accumulated, so an additional change can occur to the fluent *TotalStrokes*:

$$TotalStrokes' = TotalStrokes + 1$$

I use the prime notation here to indicate that the value of *TotalStrokes* after the change is 1 greater than the value before.

7.3 Expressing Constraints on Behavior

Every Change Means (At Least) A Stroke Intuitively, each change in location of the ball must be caused by a stroke. However, if the ball is be hit from place to place on the course without landing in the bunker, we cannot express that change in the language we have. So the stroke count can increase without a change in (expressible) location. We might suppose that the condition on location change is

$$loc(b, B) \;\&\; \neg loc'(b, B) \Rightarrow TotalStrokes' = TotalStrokes + 1$$

$$\&$$

$$\neg loc(b, B) \;\&\; loc'(b, B) \Rightarrow TotalStrokes' = TotalStrokes + 1$$

but we would be wrong. We are measuring change by comparing two states. But these two states may not represent consecutive hits of the ball. We may be comparing two states relatively far apart, say 4 or 5 strokes apart. Thus the correct condition is

$$loc(b, B) \;\&\; \neg loc'(b, B) \Rightarrow TotalStrokes' > TotalStrokes$$

$$\&$$

$$\neg loc(b, B) \;\&\; loc'(b, B) \Rightarrow TotalStrokes' > TotalStrokes$$

Expressing the Bunker Constraint We want to say that if the ball lands in the bunker, this is likely to increase the final score, but not definitely because of the chance of a birdie, as noted above. One way of saying it is that landing in the bunker increases the expected final stroke count. We need a new fluent *ExpectedScore*. The condition that landing in the bunker increases one's expected score by 1 is expressed by

$$\neg loc(b, B) \;\&\; loc'(b, B) \;\&\; TotalStrokes' = TotalStrokes + 1$$

$$\Rightarrow ExpectedScore' = ExpectedScore + 1$$

Here again, the extra conjunct in the antecedent ensures that we are comparing a change due to one stroke, not to many.

The Game Must Stop We have been assuming that *ExpectedScore* is a positive integer. There are formal ways of expressing all these conditions such that all such assumptions are explicit - for example TLA [Lam, Lam94b, Lad97], from which this notation is lifted. However, let us continue without worrying about these technical details for the moment. One important constraint on behavior is that the game stops. That means that at some point in the future, the stroke count *TotalStrokes* maintains a fixed value for ever.

Using Tense-Logical Operators Let us phrase this in terms of behavior and state. We use the tense-logical operator $\Diamond$: asserting $\Diamond A$ is to assert that *at some state in the future, A will hold.* We want to say that at some point in the future, the stroke count does not change for ever more. The "*dual*" of $\Diamond$ is the tense-logical operator $\Box$: $\Box A$ asserts that *at all states in the future, A will hold.* It is straightforward to check that

$$\Diamond A \Leftrightarrow \neg\Box\neg A$$

and

$$\Box A \Leftrightarrow \neg\Diamond\neg A$$

by referring to the behavior diagram in Figure 3.5, and imagining it going on for ever. The two equivalences can be expressed in natural language as follows. If A is true in one of the states (one can write it in to make it clear), then it cannot be the case that A is false in all the states; that is to say, that $\neg A$ is true in all the states. Similarly, if A is true in all of the states, then it cannot be the case that A is false in one of them; that is to say, it cannot be the case that $\neg A$ is true in one of them.

Expressing the Stopping Constraint To say that, at some state in the future, it will be the case from then on that the stroke count will not change, can be expressed as follows. $Box(TotalStrokes' = Totalstrokes)$ says that the stroke count remains the same for ever more. We need to say that this assertion becomes true at some state in the future. This means it needs to be prepended with $\Diamond$. So $\Diamond\Box(TotalStrokes' = TotalStrokes)$ expresses the stopping constraint. Notice that we haven't said *when* the game stops. We have not built a stopping criterion into the example yet, and won't do so.

Expressing the Accident We can now say that an accident will happen:

$$\Diamond TotalStrokes > N$$

Constraints Are Expressible Independent of the System Notice that we have been able to express the constraints unambiguously without deciding exactly in what the system consists. This phenomenon is more or less general: one can express constraints on system and environment without necessarily needing to distinguish them.

7.4 Hazard Definitions and Consequences

Hazard The reason that landing in the bunker is a hazardous situation is that the expected number of strokes is thereby increased. If one has already birdied five times, then landing in a bunker and expecting to lose a stroke is not particularly problematic, or hazardous, because one then expects merely to finish four under instead of five under. I suggest that landing in the bunker becomes hazardous when the expected number of strokes increases to above the limit N. In fact, bunker or no bunker, this is a hazard, and remains a hazard until one birdies to get it down again. So we might try to express being in the hazardous situation by the state predicate $\mathit{ExpectedScore} > N$.

Hazard and World State There are only two objects (one fluent and one integer) mentioned in the state predicate that expresses a hazard. Neither of these objects belongs to the system as we have so far conceived it, or the environment as we have so far conceived it. There are the following possibilities:

- If integers and the fluent belong to the system, then the hazard definition is a system predicate; thus a Hazard-1-type definition.
- If integers and the fluent belong to the environment, then the hazard definition is an environment predicate; thus a Hazard-2-type definition.
- If one belongs to the system and the other to the environment, then the hazard definition is a hybrid predicate; thus a Hazard-4-type definition.

Is An Accident Is Inevitable? From the state predicate $\mathit{ExpectedScore} > N$ it does not follow that an accident is inevitable. However, we may imagine that *ExpectedScore* is really the *expected score*, and if we get to within one stroke of finishing and the expected score is still greater than N, then the total we have already, *TotalScore*, must be already greater than or equal to N. So if the expected score rises to above N, and remains there until the game finishes, then an accident is inevitable. That is,

$$\mathit{ExpectedScore} > N \;\&\; \Box(\mathit{ExpectedScore} > N)$$

$$\Rightarrow \Diamond(\mathit{TotalScore} > N)$$

That is, an accident is inevitable if the predicate

$$\mathit{ExpectedScore} > N \;\&\; \Box(\mathit{ExpectedScore} > N)$$

ever becomes true. A technical point: in tense logic as used in engineering, the statement $\Box A \Rightarrow A$ is taken to be an axiom, for any statement A. Hence we may write the statement

$$\mathit{ExpectedScore} > N \;\&\; \Box(\mathit{ExpectedScore} > N)$$

as its tense-logical equivalent

$$\Box(\mathit{ExpectedScore} > N)$$

without loss of expressiveness.

Summary of the Hazard Definitions We may summarise the situation with regard to the use of hazard definitions as follows.

- if we use the hazard definition $\Box(\mathit{ExpectedScore} > N)$ then
  - We may use a Hazard-1 definition only if both *ExpectedScore* and N belong to the system;
  - We may use a Hazard-2 definition only if both *ExpectedScore* and N belong to the environment;
  - If one of *ExpectedScore* and N belongs to the system and the other to the environment, we may use Hazard-4;
- Consider the hazard definition $(\mathit{ExpectedScore} > N)$. An accident is not inevitable from a state satisfying $(\mathit{ExpectedScore} > N)$ unless it also satisfies $\Box(\mathit{ExpectedScore} > N)$. To use a Hazard-1 or Hazard-2 definition, one of these predicates would have to be a system predicate and the other an environment predicate (recall that according to these definitions, and accident is inevitable if a hazard state (system or environment respectively) *coupled with* a complimentary state (of environment or system, respectively) leads inevitably to an accident. Since both predicates mention the same objects, they are either both system or both environment, and cannot be one and the other. Hence a Hazard-1 or Hazard-2 definition cannot be used. A Hazard-4 definition also requires inevitability. Hazard-3 is the only definition which requires just increased likelihood. Hence if the hazard definition is taken to be $(\mathit{ExpectedScore} > N)$, a Hazard-3 definition must be used.

Hazard Definition Type Depends on What's In The System We may conclude that which type of hazard definition is used depends on what one considers to be part of the system and what not.

- If one insists on using a Hazard-1 definition, then one *must* include *ExpectedScore* and integers (at least, N) as part of the system.
- If one insists on using a Hazard-2 definition, then one *must* include *ExpectedScore* and integers (at least, N) as part of the environment.
- One cannot use a Hazard-4 definition.
- A Hazard-3 definition can be used which is logically simpler (it does not include the tense-logical operator $\Box$), no matter where *ExpectedScore* and N are chosen to belong.