MODULE *Prophecy*

This module defines operators used in adding a prophecy variable, as described in the paper "Auxiliary Variables in TLA+". To add a prophecy variable to a specification, we first choose a disjunctive representation of its next-state action–a way of decomposing the action into subactions that is explained in Section 3.2 of the paper. The operators defined here are explained in Sections 4.4 and 4.5 of the paper.

We first define $PartialInjections(U, V)$ to be the set partial injections from U to V. It consists of all injective (one-one) functions from a subset of U into V. This definition is used below.

```
PartialInjections(U, V) ≜
  LET PartialFcns ≜ UNION {[D → V] : D ∈ SUBSET U}
  IN  {f ∈ PartialFcns : ∀ x, y ∈ DOMAIN f : (x ≠ y) ⇒ (f[x] ≠ f[y])}
```

The value of a prophecy variable p is always a function from some set *Dom* such that, for every $d \in Dom$ the value of $p[d]$ represents a prediction associated with d. This module is meant to be instantiated with the constants *Pi*, *Dom*, and *DomPrime* replaced as follows.

Pi: Instantiated by a constant set of possible predictions.

Dom: Instantiated by a state function whose value is the domain of p.

DomPrime: Instantiated by Dom' – that is, the primed version of the state function with which *Dom* is instantiated.

```
CONSTANTS Pi, Dom, DomPrime
```

To add a prophecy variable to a spec, we have to define three things for every subaction A of a disjunctive representation of its next-state action: An operator *Pred* that takes a single argument, and a two expressions *PredDom* and *DomInj*. To add the prophecy variable p, we replace the subaction A by action *Ap* defined to equal

$A \land Pred(p) \land \ (p' \in NewPSet(p, DomInj, PredDom))$

where *NewPSet* is defined below. Here are the meanings of these three defined quantities:

Pred

$Pred(p)$ is the prediction that the value of p is making about action A. For example, if $Pred(p)$ predicts that A will set the value of x to $p[d]$ for some particular d in *Dom*, then $Pred(p)$ equals $x' = p[d]$.

If A occurs in some context, then *Pred* may need additional arguments. For example, if A occurs in $\forall i \in \ I : A$, then the prediction p is making about A may depend on both p and i. You may then have to define the operator $Pred(p, i)$. In that case, an operator that is defined with a one-argument operator *Pred* should be given the argument LAMBDA $q : Pred(q, i)$.

PredDom

The subset of *Dom* consisting of the elements d for which $Pred(p)$ may depend on the value of $p[d]$.

DomInj

A partial injection from *Dom* to Dom' describing the correspondence between predictions made by p and those made by p'. More precisely, the prediction p makes for d in Dom' is the same prediction it was making for c in *Dom* if and only if $DomInj[c] = d$. For example, suppose the value of p is a sequence of predictions being made about action A, element i of p making a prediction about the i th next A step. Then executing *Ap* removes the first element of p, setting p' to $Tail(p)$. (Some other action will append new predictions to p.) Then the prediction made by $p'[j]$ is the same as the prediction made by $p'[j+1]$, so *DomInj* should be defined to equal $[i \in 2 \,..\, Len(p) \mapsto i-1]$.

We now define conditions that the definitions of *Pred*, *PredDom*, and *DomInj* for the actions A should satisfy to ensure that they produce a prophecy variable— one that allows all the same behaviors of the original variables.

$$ExistsGoodProphecy(Pred(\_)) \triangleq \exists\, q \in [Dom \rightarrow Pi] : Pred(q)$$

Asserts that there exists some possible prediction p that make $Pred(p)$ true–a requirement that the action *Ap* allows all the possibilities that action A does.

$$IsDomInj(DomInj) \triangleq DomInj \in PartialInjections(Dom, DomPrime)$$

This is an obvious type-correctness condition for *DomInj*.

$$IsPredDom(PredDom, Pred(\_)) \triangleq$$

This condition asserts that *PredDom* contains all the elements d of *Dom* such that $p[d]$ makes a predication about the action–that is, such that $p[d]$ affects the value of $Pred(p)$. It's essential that if d makes a prediction about the action and *DomInj* implies that the prediction made by $p[d]$ corresponds to the prediction made by $p'[c]$, then the definition of *NewPSet* must ensure that $p'[c]$ can assume any value in *Pi*. Allowing a prediction to be used twice can rule out behaviors allowed by the original specification. It's *OK* if *PredDom* also contains elements d of *Dom* that don't make predictions about the action. It can't hurt to let $p'[c]$ assume any value in *Pi*, replacing a prediction that hasn't been used with a new predictiion.

$$\begin{aligned}
&\land\ PredDom \subseteq Dom \\
&\land\ \forall\, q, r \in [Dom \rightarrow Pi] : \\
&\qquad (\forall\, d \in PredDom : q[d] = r[d]) \Rightarrow (Pred(q) = Pred(r))
\end{aligned}$$

$$NewPSet(p, DomInj, PredDom) \triangleq$$

This is the set of all possible new values of p that can result after an action is taken that satisfies the prediction made by p. This set must allow $p'[d]$ to assume any value in *Pi* if the element d in Dom' either (a) corresponds under *DomInj* to an element c in *PredDom* (and hence $p[d]$ may have made a prediction about action A that was used) or (b) does not correspond under *DomInj* to any element of *Dom* (and hence is making a new prediction).

$$\begin{aligned}
&\{q \in [DomPrime \rightarrow Pi] : \\
&\quad \forall\, d \in (\textsc{domain}\ DomInj) \setminus PredDom : q[DomInj[d]] = p[d]\}
\end{aligned}$$

We now define the operator *ProphAction* so that the subaction *Ap* of the new specification that replaces subaction A of the original specification equals

$ProphAction(A, p, p', DomInj, PredDom, Pred)$

Remember that *Pred* must be replaced by a LAMBDA expression (or by the equivalent defined one-argument operator) if it has more than just the "p argument".

$$\begin{aligned}
&ProphAction(A, p, pPrime, DomInj, PredDom, Pred(\_)) \triangleq \\
&\quad \land\ A \\
&\quad \land\ Pred(p)
\end{aligned}$$

$\wedge\ pPrime \in NewPSet(p,\ \ DomInj,\ PredDom)$

We also combine the requirements above on *Pred*, *DomInj*, and *PredDom* for a subaction A into a single condition. For each subaction A, the original specification should satisfy the property

$\Box[ProphCondition(A,\ DomInj,\ PredDom,\ Pred)]\_vars$

where *vars* is the tuple of all its variables.

$$
\begin{array}{l}
ProphCondition(A,\ DomInj,\ PredDom,\ \ Pred(\_)) \ \triangleq \\
\quad A \Rightarrow \ \wedge\ ExistsGoodProphecy(Pred) \\
\qquad\qquad \wedge\ IsDomInj(DomInj) \\
\qquad\qquad \wedge\ IsPredDom(PredDom,\ Pred)
\end{array}
$$

We now make two definitions for convenience in using the operators defined here:

EmptyFcn : The unique function whose domain is the empty set.

IdFcn(S) : The function in $[S \rightarrow S]$ that maps every element of S to itself.

$EmptyFcn \ \triangleq\ \langle\rangle$

$IdFcn(S)\ \ \triangleq\ [i \in S \mapsto i]$

\* Modification History
\* Last modified *Wed Oct* 19 08:34:27 *PDT* 2016 by *lamport*
\* Created *Fri Sep* 16 06:52:33 *PDT* 2016 by *lamport*