Exercise 2: The World According to Chronos

Archaeologists recently discovered a manuscript from an ancient Greek natural philosopher named Chronos, whose work was previously unknown. Chronos was ahead of his time, but his theoretical ideas were tragically flawed. You have been asked to formalize his theories in PVS.

1. The file `chronos.pvs` contains two PVS theories: one is the `vectors` theory already presented, and the other is a mostly empty theory named `chronos`. Bring up the file using the command `M-x ff` (find file). Go ahead and typecheck it. You will be building up the theory `chronos` incrementally in the following steps. Typecheck it again after each item is added. Keep correcting any specification errors until the theory typechecks cleanly at each step.

2. Chronos used a three-axis coordinate system to model the physical world. His axes were labeled α, β and γ. Introduce an enumeration type called something like `chr_axis` to represent these axes. Import the theory `vectors` with this type as a parameter.

3. In his studies Chronos used a simple calendar that was nothing more than a numbering of the days. Day One he considered to be the day of his own birth. Day Omega was his hypothesized end date for the world as we know it, which he estimated would occur in one million days. Introduce a constant for Day Omega, then declare a type such as `calendar_day` to model the Chronosian calendar using a suitable predicate subtype of `nat`.

4. Remarkably, Chronos hypothesized a law of universal gravitation long before Newton did. In a bizarre twist, however, he believed the Earth's gravitational acceleration "constant" increased in stepwise fashion once a day at midnight. Declare a type `gravity_fn` to represent functions from `calendar_day` to `real`, which can model the time-varying acceleration due to gravity.

5. Introduce a constant for the number of seconds in a day. Declare a type for the time of day expressed as seconds past midnight. This type should be a predicate subtype of `real`.

6. Declare a record type representing the concept of "point in time" that contains a field for calendar day and a field for time of day.

7. Define a predicate (boolean-valued function) to test whether one point in time precedes another, i.e., is less than or equal using a lexicographic ordering.

```
precedes(x, y: point_in_time): bool = <expression>
```

8. Define a function that takes a point-in-time value y and another value x that precedes y, then computes their time difference in seconds. Note that the type of x depends on the value of y:

```
time_diff(y: point_in_time,
          x: {t: point_in_time | precedes(t, y)}): real =
    <expression>
```

9. Declare a type that represents vectors in the Chronosian coordinate system. Use the imported `vectors` theory and specialize it with your axis type declared in step 2.

10. Using the vector type from step 9, define a function that computes the final position of a moving body in Chronosian coordinates when given initial position and velocity vectors. Give the function four arguments: a point in time x, a later point in time y, and the position and velocity vectors representing the state at time x. Assume linear motion and no net force acting on the body. Use the vector operations defined in the `vectors` theory and the time difference function from step 8. (Recall from physics the vector equation $\vec{p_2} = \vec{p_1} + \vec{v}t$.)

```
final_position(x: point_in_time,
               y: {t: point_in_time | precedes(x, t)},
               position, velocity: chr_vector): chr_vector =
    <expression>
```

11. The Chronosian Principle of Daily Gravitational Progression holds that the gravitation "constant" is a monotonically increasing function of calendar day. Incidentally, Chronos also theorized that gravity eventually would become so strong that the Earth would collapse (on Day Omega) into what we now call a black hole. Declare a function of type `gravity_fn` g and express the montonicity principle as a constraint in a predicate subtype.

```
g: {f: gravity_fn | <constraint>}
```

12. Define a function that takes a point-in-time x and a duration t, and calculates, according to Chronosian theory, the speed a falling object would have attained after t seconds if dropped at point in time x. (Recall from physics that the speed change Δv experienced under constant acceleration a is given by at.)

Restrict x to be earlier than Day Omega and t to be less than one day. Remember what happens to gravity at midnight. Ignore the slight increase in weight as the object falls, the effects of wind resistance, etc. (Hint: split the duration into two portions that lie on either side of midnight following the initial time x, then use LET to name the two times.)

```
final_speed(x: {T: point_in_time | T`day < Omega},
            t: {d: real | 0 <= d & d < sec_per_day}): real =
      LET delta_T_before =
            <expression>,
          delta_T_after =
            <expression>
      IN
          <expression>
```

13. After successfully typechecking your theory, look over any TCCs that are generated by issuing the command `M-x show-tccs`. If any of them look untrue, try to determine why and revise your theory accordingly.

14. Make up any other interesting concepts from the world according to Chronos and express them in PVS.

```
%%=============================================================================
%%         Generic vector operations on real elements.
%%=============================================================================


vectors [index_type: TYPE]: THEORY
BEGIN

vector:         TYPE = [index_type -> real]

i,j,k:          VAR index_type
a,b,c:          VAR real

U,V:            VAR vector

zero_vector:       vector = LAMBDA i: 0

vector_sum(U, V):  vector = LAMBDA i: U(i) + V(i)

vector_diff(U, V): vector = LAMBDA i: U(i) - V(i)

scalar_mult(a, V): vector = LAMBDA i: a * V(i)

END vectors


%%=============================================================================
%%            The World According to Chronos  (Exercise 2)
%%=============================================================================

chronos: THEORY
BEGIN

%% Insert new declarations in this theory.

%% Stubs for some function declarations are provided in
%% the file chronos.pvs.

END chronos
```