

sequentially wait-for all

>>>file: Cholesterol-tests

(:

Name: Cholesterol-tests

Status:

```
(plan name="Cholesterol-tests"
  (plan-body (subplans type="sequentially"
    (wait-for (all))
    (ask (parameter-ref name="total-
cholesterol")
      (time-out (now)))
    (ask (parameter-ref name="HDL-
cholesterol")
      (time-out (now)))
    (ask (parameter-ref
name="triglycerids")
      (time-out (now))))))
:)
```

enrich patient-data, time, plan-state with

procedures

Cholesterol-tests# : patient-data, nat,
plan-state

nonfunctional indeterministic;

variables

total-cholesterol : bool;

HDL-cholesterol : bool;

triglycerids : bool;

declaration

```

Cholesterol-tests#(var pd, time, Cholesterol-
tests-state)
begin
  var total-cholesterol = [?], HDL-cholesterol
= [?], triglycerids = [?]
  in
  begin
    total-cholesterol := [?];
    pd := set-total-cholesterol(pd, total-
cholesterol);
    var time0 = time, dt = [?] in await
time0 + dt \le time;

    HDL-cholesterol := [?];
    pd := set-HDL-cholesterol(pd, HDL-
cholesterol);
    var time0 = time, dt = [?] in await
time0 + dt \le time;

    triglycerids := [?];
    pd := set-triglycerids(pd,
triglycerids);
    var time0 = time, dt = [?] in await
time0 + dt \le time;

    Cholesterol-tests-state := completed
  end
end;

end enrich

```

sequentially wait-for one wait-for-
optional-subplans

```

>>>file: Diabetes-Mellitus-Type-2
(:
Name: Diabetes-Mellitus-Type-2
Status:

    (plan name="Diabetes-Mellitus-Type-2"
      (plan-body (subplans type="sequentially"
                          wait-for-optional-
subplans="yes"
                    (wait-for (one))
                    (plan-activation (plan-schema
name="Diagnostics"))
                    (plan-activation (plan-schema
name="Policy")))))
:))

enrich Diagnostics, Policy, patient-data,
time, plan-state with

procedures
  Diabetes-Mellitus-Type-2# : patient-data,
nat, plan-state
  nonfunctional indeterministic;

variables
  Diagnostics-state : plan-state;
  Policy-state : plan-state;

declaration

Diabetes-Mellitus-Type-2#(var pd, time,
Diabetes-Mellitus-Type-2-state)
begin
  var
    Diagnostics-state = inactive,

```

```

        Policy-state = inactive
    in
    begin
        Diagnostics#(; pd, time, Diagnostics-
state);
        Policy#(; pd, time, Policy-state);

        if      Diagnostics-state = completed
            or Policy-state = completed
        then Diabetes-Mellitus-Type-2-state :=
completed
        else Diabetes-Mellitus-Type-2-state :=
aborted
        end
    end;

end enrich

```

sequentially wait-for certain plan
wait-for-optional-subplans

```

>>>file: Diagnostics
(:
Name: Diagnostics
Status:

(plan name="Diagnostics"
  (plan-body (subplans type="sequentially"
                    wait-for-optional-
subplans="yes"
                (wait-for (static-plan-pointer plan-
name="Anamnesis"))

```

```

        (plan-activation (plan-schema
name="Anamnesis"))
        (plan-activation (plan-schema
name="Glucose-determination"))
        (plan-activation (plan-schema
name="Risk-inventory")))))
: )

```

```

enrich Anamnesis, Glucose-determination, Risk-
inventory,
        patient-data, time, plan-state with

```

procedures

```

Diagnostics# : patient-data, nat, plan-state
nonfunctional indeterministic;

```

variables

```

Anamnesis-state : plan-state;
Glucose-determination-state : plan-state;
Risk-inventory-state : plan-state;

```

declaration

```

Diagnostics#(var pd, time, Diagnostics-state)
begin
    var
        Anamnesis-state = inactive,
        Glucose-determination-state = inactive,
        Risk-inventory-state = inactive
    in
    begin
        Anamnesis#(; pd, time, Anamnesis-state);
        Glucose-determination#(; pd, time,
Glucose-determination-state);
        Risk-inventory#(; pd, time, Risk-

```

```
inventory-state);  
  if Anamnesis-state = completed  
  then Diagnostics-state := completed  
  else Diagnostics-state := aborted  
  end  
end;  
  
end enrich
```

any-order wait-for all

```
>>>file: Glucose-determination  
(:  
Name: Glucose-determination  
Status: time-annotation in plan-activation  
needs discussion  
  
  (plan name="Glucose-determination"  
    (conditions (filter-precondition (  
      "equal" "glucose-  
determination-needed" "true")))))  
    (plan-body (subplans type="sequentially"  
      (wait-for (all))  
      (subplans type="any-order"  
        (wait-for (one))  
        (plan-activation  
          (plan-schema name="Fasting-  
glucose-test-manual"))  
        (plan-activation  
          (plan-schema name="Non-fasting-  
glucose-test-manual"))  
        (if-then-else (simple-condition
```

```

(comparison type="equal"
      (left-hand-side (parameter-ref
name="glucose-evaluation"))
      (right-hand-side (qualitative-
constant value="DMT2"))))
      (then-branch (plan-activation
                    (plan-schema
name="Fasting-glucose-test"
      (time-annotation (time-range
(starting-shift
                        (earliest (numerical-
constant scale="time"
unit="d" value="2"))
                        (latest (numerical-
constant scale="time"
unit="d" value="7"))))
      (references (plan-state-
transition direction="leave"
                    instance-
type="first" state="completed"
                    (plan-pointer (static-
plan-pointer
                                plan-
name="Fasting-glucose-test-manual"))
                    (plan-state-transition
direction="leave"
                                instance-
type="first" state="completed"
                                (plan-pointer (static-
plan-pointer
                                plan-name="Non-fasting-
glucose-test-manual"))))))))))))
: )

```

```
enrich Fasting-glucose-test-manual, Non-  
fasting-glucose-test-manual,  
      Fasting-glucose-test, patient-data,  
time, plan-state with
```

```
procedures
```

```
  Glucose-determination# : patient-data, nat,  
plan-state  
  nonfunctional indeterministic;
```

```
variables
```

```
  Fasting-glucose-test-manual-state : plan-  
state;  
  Non-fasting-glucose-test-manual-state :  
plan-state;  
  Fasting-glucose-test-state : plan-state;
```

```
declaration
```

```
Glucose-determination#(var pd, time, Glucose-  
determination-state)  
begin  
  var  
    Fasting-glucose-test-manual-state =  
inactive,  
    Non-fasting-glucose-test-manual-state =  
inactive,  
    Fasting-glucose-test-state = inactive  
  in  
  begin  
    await glucose-determination-needed;  
    (: any-order wait-for all :)  
    begin  
      (: any-order wait-for one :)
```



```

                pbreak
                Fasting-glucose-test-manual#( ;
pd, time,
                Fasting-
glucose-test-manual-state);
                ||a Non-fasting-glucose-test-
manual#( ; pd, time,
                Non-fasting-
glucose-test-manual-state);
                if      Fasting-glucose-test-manual-
state = completed
                    or Non-fasting-glucose-test-
manual-state = completed
                end
                ||a if glucose-evaluation = DMT2
                then
                    begin
                        while true do
                            var time0 = time in await 2 \le
time - time0
                                and
time - time0 \le 7;
                            Fasting-glucose-test#( ; pd, time,
Fasting-glucose-test-state);
                            end
                            if      (Fasting-glucose-test-manual-state
= completed
                                or Non-fasting-glucose-test-manual-
state = completed)
                                and (Fasting-glucose-test-state =
completed
                                    or pd.glucose-evaluation unequal
DMT2)
                            then Glucose-determination-state :=
completed

```

```
    else Glucose-determination-state := aborted
  end
end;

end enrich
```

any-order wait-for one

```
>>>file: Albumin-test
(:
Name: Albumin-test
Status:
```

```
    (plan name="Albumin-test"
      (plan-body (subplans type="any-order"
        (wait-for (one))
        (plan-activation (plan-schema
name="Albumin-test-manual"))
        (plan-activation (plan-schema
                          name="Albumin-
creatinin-ratio-test-manual")))))
:)
```

```
enrich Albumin-test-manual, Albumin-creatinin-
ratio-test-manual,
      patient-data, time, plan-state with
```

procedures

```
  Albumin-test# : patient-data, nat, plan-
state
  nonfunctional indeterministic;
```

```

variables
    Albumin-test-manual-state : plan-state;
    Albumin-creatinin-ratio-test-manual-state :
plan-state;

declaration

Albumin-test#(var pd, time, Albumin-test-
state)
begin
    var
        Albumin-test-manual-state = inactive,
        Albumin-creatinin-ratio-test-manual-state
= inactive
    in
        begin
            pbreak
                Albumin-test-manual#(; pd, time,
Albumin-test-manual-state);
                ||a Albumin-creatinin-ratio-test-
manual#(; pd, time,
                    Albumin-
creatinin-ratio-test-manual-state);
                if Albumin-test-manual-state =
completed
                    or Albumin-creatinin-ratio-test-manual-
state = completed
                    if Albumin-test-manual-state =
completed
                        or Albumin-creatinin-ratio-test-manual-
state = completed
                        then Albumin-test-state := completed
                        else Albumin-test-state := aborted
                    end
                end;
end;

```

```
end enrich
```

```
parallel wait-for none & on-abort &  
cyclical
```

```
>>>file: Treatments-and-Controls
```

```
(:
```

```
Name: Treatments-and-Controls
```

```
Status:
```

```
    (plan name="Treatments-and-Controls"  
      (conditions (complete-condition (simple-  
condition  
      (comparison type="equal"  
        (left-hand-side (variable-ref  
name="alive"))  
        (right-hand-side (qualitative-  
constant value="false")))))  
      (plan-body (subplans type="parallel"  
        (wait-for (none))  
        (plan-activation (plan-schema  
name="Non-insulin-DMT2-treatments")  
          (on-abort (plan-activation (plan-  
schema  
name="Insulin-DMT2-treatments"))))  
          (plan-activation (plan-schema  
name="Treatment-  
of-CV-disease-risk-factors"))  
          (cyclical-plan (cyclical-plan-body  
            (plan-activation (plan-schema  
name="Quarterly-control")))  
            (repeat-specification (retry-delay
```

```

                (minimum (numerical-constant
scale="time" unit="mon"
value="2.5"))
                (maximum (numerical-constant
scale="time" unit="mon"
value="3.5"))))
        (cyclical-plan (cyclical-plan-body
            (plan-activation (plan-schema
name="Annual-control"))
            (repeat-specification (retry-delay
                (minimum (numerical-constant
scale="time" unit="mon"
value="11.5"))
                (maximum (numerical-constant
scale="time" unit="mon"
value="12.5"))))))))
    :)

```

```

enrich Non-insulin-DMT2-treatments, Insulin-
DMT2-treatments,
    Treatment-of-CV-disease-risk-factors,
Quarterly-control,
    Annual-control,
    patient-data, time, plan-state with

```

procedures

```

    Treatments-and-Controls# : patient-data,
nat, plan-state
    nonfunctional indeterministic;

```

variables

```

    Non-insulin-DMT2-treatments-state : plan-
state;
    Insulin-DMT2-treatments-state : plan-state;
    Treatment-of-CV-disease-risk-factors-state :
plan-state;
    Quarterly-control-state : plan-state;
    Annual-control-state : plan-state;

```

declaration

```

Treatments-and-Controls#(var pd, time,
Treatments-and-Controls-state)
begin
    var
        Non-insulin-DMT2-treatments-state =
inactive,
        Insulin-DMT2-treatments-state = inactive,
        Treatment-of-CV-disease-risk-factors-state
= inactive,
        Quarterly-control-state = inactive,
        Annual-control-state = inactive
    in
    begin
        pbreak
            Non-insulin-DMT2-treatments#(; pd,
time,
                                Non-
insulin-DMT2-treatments-state);
            (: on-abort :)
            if Non-insulin-DMT2-treatments-state
= aborted
            then Insulin-DMT2-treatments#(; pd,
time,
Insulin-DMT2-treatments-state)

```

```

        ||s Treatment-of-CV-disease-risk-
factors#(; pd, time,
                                Treatment-of-CV-
disease-risk-factors-state);
        ||s while true do                !!!!!!!!!!!!!!!
            (: time axis is 1/10 month here
              - will be changed later to
days, e.g. :)
            var time0 = time in await 25 \le
time - time0
                                and
time - time0 \le 35
            Quarterly-control#(; pd, time,
Quarterly-control-state);
        ||s Annual-control#(; pd, time, Annual-
control-state);
            while true do
                var time0 = time in await 115 \le
time - time0
                                and
time - time0 \le 125;
                if not pd.alive;
                if not pd.alive
                then Treatments-and-Controls-state :=
completed
                else Treatments-and-Controls-state :=
aborted
                end
            end;
end enrich

```

unordered, wait-for certain plan,
retry-aborted-subplans

>>>file: Policy

(:

Name: Policy

Status:

```
(plan name="Policy"
  (conditions (filter-precondition
    (simple-condition
      (comparison type="equal"
        (left-hand-side (variable-ref
          name="glucose-evaluation"))
        (right-hand-side (qualitative-
          constant value="DMT2")))))
    (plan-body (subplans type="sequentially"
      (wait-for (all))
      (plan-activation (plan-schema
        name="Education-DMT2"))
      (subplans retry-aborted-
        subplans="yes" type="unordered"
        (wait-for (static-plan-pointer
          plan-
            name="Treatments-and-Controls"))
          (plan-activation (plan-schema
            name="Treatments-and-Controls"))
          (plan-activation
            (plan-schema name="Policy-for-
              concurrent-diseases"))
          (plan-activation
            (plan-schema name="Policy-for-
              hypoglycemic-coma"))
          (plan-activation
            (plan-schema name="Policy-for-
              consultation"))
          (plan-activation
```



```
        (plan-schema name="Policy-for-
chiropracist-referral"))
      (plan-activation
        (plan-schema name="Policy-for-
nurse-referral"))))))
:)
```

```
enrich Education-DMT2, Treatments-and-
Controls,
      Policy-for-concurrent-diseases,
      Policy-for-hypoglycemic-coma, Policy-
for-consultation,
      Policy-for-chiropracist-referral,
Policy-for-nurse-referral,
      patient-data, time, plan-state with
```

procedures

```
Policy# : patient-data, nat, plan-state
nonfunctional indeterministic;
```

```
Education-DMT2# : patient-data, nat, plan-
state
nonfunctional indeterministic;
```

variables

```
Education-DMT2-state : plan-state;
Treatments-and-Controls-state : plan-state;
Policy-for-concurrent-diseases-state : plan-
state;
Policy-for-hypoglycemic-coma-state : plan-
state;
Policy-for-consultation-state : plan-state;
Policy-for-chiropracist-referral-state :
plan-state;
Policy-for-nurse-referral-state : plan-
```

state;

declaration

Policy#(var pd, time, Policy-state)

begin

var

Education-DMT2-state = inactive,

Treatments-and-Controls-state = inactive,

Policy-for-concurrent-diseases-state =
inactive,

Policy-for-hypoglycemic-coma-state =
inactive,

Policy-for-consultation-state = inactive,

Policy-for-chiropracist-referral-state =
inactive,

Policy-for-nurse-referral-state = inactive

in

begin

await glucose-evaluation = DTM;

Education-DMT2#(; pd, time, Education-
DMT2-state);

begin

break !!!!!!!!!!!!!

Treatments-and-Controls#(; pd, time,
Treatments-and-Controls-state)

||s while Policy-for-concurrent-
diseases-state unequal completed

do Policy-for-concurrent-diseases#(
pd, time,

Policy-
for-concurrent-diseases-state);

||s while Policy-for-hypoglycemic-coma-
state unequal completed

do Policy-for-hypoglycemic-coma#(;

```

pd, time,
                                Policy-
for-hypoglycemic-coma-state);
    ||s while Policy-for-consultation-state
unequal completed
        do Policy-for-consultation#(; pd,
time,
                                Policy-
for-consultation-state);
    ||s while Policy-for-chiropracist-
referral-state unequal completed
        do Policy-for-chiropracist-
referral#(; pd, time,
                                Policy-
for-chiropracist-referral-state);
    ||s while Policy-for-nurse-referral-
state unequal completed
        do Policy-for-nurse-referral#(; pd,
time,
                                Policy-
for-nurse-referral-state);
    end

    if      Education-DMT2-state = completed
        and Treatments-and-Controls-state =
completed
    then Policy-state := completed
    else Policy-state := aborted
    end
end;

(:
Name: Education-DMT2
Status:

```

```

    (plan name="Education-DMT2"
          title="Give information about the
most important aspects of DM"
          (plan-body (user-performed)))
: )

```

```

Education-DMT2#(var pd, time, Education-DMT2-
state)
begin
  skip;
  var time0 = time, dt = [?] in await time0 +
dt \le time;
  Education-DMT2-state := completed
end;

end enrich

```

unordered wait-for certain plan

```

>>>file: SU-derivative-plus-metformin-
treatment
( :
Name: SU-derivative-plus-metformin-treatment
Status: pseudo-code

```

```

    (plan name="SU-derivative-plus-metformin-
treatment"
          ...
          (subplans type="unordered"
            (wait-for (static-plan-pointer
                      plan-name="Find-
antidiabetic-doses"))
            (plan-activation (plan-schema

```

```

name="Find-
antidiabetic-doses"
    (argument-value ...)
    (plan-activation (plan-schema

name="Check-for-antidiabetic-problems"
    (argument-value ...)
: )

enrich Initialise-drug-doses, Find-
antidiabetic-doses,
    Check-for-antidiabetic-problems,
    patient-data, time, plan-state with

procedures
    SU-derivative-plus-metformin-treatment# :
patient-data, nat, plan-state
    nonfunctional indeterministic;

variables
    drug-name : bool;
    Initialise-drug-doses-state : plan-state;
    Find-antidiabetic-doses-state : plan-state;
    Check-for-antidiabetic-problems-state :
plan-state;

declaration

SU-derivative-plus-metformin-treatment#(var
pd, time,
                                SU-derivative-plus-
metformin-treatment-state)
begin
    var drug-name = [?],
        Initialise-drug-doses-state = inactive,

```

```

        Find-antidiabetic-doses-state = inactive,
        Check-for-antidiabetic-problems-state =
inactive
    in
    begin
        ...
        pbreak
            Find-antidiabetic-doses#(; pd, time,
Find-
antidiabetic-doses-state)
        ||s Check-for-antidiabetic-problems#(;
pd, time,
Check-for-
antidiabetic-problems-state);
        if Find-antidiabetic-doses-state =
completed;
        (: do not
wait for optional subplans

    OK

    :)
        if Find-antidiabetic-doses-state =
completed
            and Initialise-drug-doses-state =
completed
            then SU-derivative-plus-metformin-
treatment-state := completed
            else SU-derivative-plus-metformin-
treatment-state := aborted
            end
        end;

end enrich

```

unordered wait-for-optional-subplans
wait-for none

```
>>>file: Treatment-of-CV-disease-risk-factors  
(:  
Name: Treatment-of-CV-disease-risk-factors  
Status:
```

```
    (plan name="Treatment-of-CV-disease-risk-  
factors"  
      (plan-body (subplans type="unordered"  
wait-for-optional-subplans="yes"  
        (wait-for (none))  
        (plan-activation (plan-schema  
name="Smoking-advice"))  
        (plan-activation (plan-schema  
name="Hypertension-treatment"))  
        (plan-activation (plan-schema  
name="Cholesterol-treatment"))  
        (plan-activation (plan-schema  
name="Microalbuminuria-treatment")))))  
:)
```

enrich Microalbuminuria-treatment, patient-
data, time, plan-state with

procedures

```
Treatment-of-CV-disease-risk-factors# :  
patient-data, nat, plan-state  
nonfunctional indeterministic;
```

variables

```
Smoking-advice-state : plan-state;
```

```

Hypertension-treatment-state : plan-state;
Cholesterol-treatment-state : plan-state;
Microalbuminuria-treatment-state : plan-
state;

declaration

Treatment-of-CV-disease-risk-factors#(var pd,
time,
                                Treatment-of-CV-
disease-risk-factors-state)
begin
    var
        Smoking-advice-state = inactive,
        Hypertension-treatment-state = inactive,
        Cholesterol-treatment-state = inactive,
        Microalbuminuria-treatment-state =
inactive
    in
        begin
            (: unordered, wait for none, wait for
optional subplans :)
            Smoking-advice#(var pd, time, Smoking-
advice-state)
            ||s Hypertension-treatment#(var pd, time,
Hypertension-treatment-state)
            ||s Cholesterol-treatment#(var pd, time,
Cholesterol-treatment-state)
            ||s Microalbuminuria-treatment#(var pd,
time,
Microalbuminuria-treatment-state);
            Treatment-of-CV-disease-risk-factors-state
:= completed
        end

```


end;