Translating IDEF3 to PSL*

Mihai Ciocoiu
Dept. of Computer Science
and Institute for Systems Research
University of Maryland
College Park, MD 20742
mihaic@cs.umd.edu

Abstract

This document describes the process of integrating IDEF3 and PSL. The EPIF like frame representation developed for representing IDEF3 schematics is introduced, together with the compilation rules for the various IDEF3 elements. The appendix contains a full example of the use of the translator for the Camile scenario.

*This work was partially supported by NIST contract 70NANB6H0147 — Process Representation Studies.

Contents

1 Introduction

This document describes the process of integrating IDEF3[3] and PSL. Section 2 introduces the EPIF[2] like frame representation developed for representing IDEF3 schematics. Then, in Section 3 the PSL extensions that were necessary for representing IDEF3 concepts are introduced, together with their definitions.

The translation process is based on a set of *compilation rules* that associate with each IDEF3 concept a KIF[1] sentence expressing its semantics into PSL. These compilation rules are introduced in Section 4.

The actual implementation of the translator is presented in Section 5, together with an example of its use to translate an IDEF3 schematic representation of the Camile[4] scenario into PSL.

2 The IDEF3 language

IDEF3[3] is a graphical language designed for capturing information about the objects and processes involved in a system. It offers both process-centered and an object-centered perspectives, and it includes the ability to capture and structure descriptions of how a system works from multiple viewpoints.

Unfortunately, it turns out that apart from its graphical elements, there is no standard textual representation for its elements.

A preliminary such encoding, specified by Chris Menzel at Knowledge Based Systems, Inc. and used in [5] was selected as a basis for building a representation language rich enough to capture the IDEF3 elements. Frames were provided and compilation rules defined, in the stile of EPIF[2] for the various kinds of IDEF3 junctions and links.

The two major differences from EPIF is that our representation reifies activity types and that explicit link constructs are used as opposed to `:successors` fields in EPIF. This change was imposed by the fact that there are several kinds of expressing successor relationships in IDEF3 with slighty different semantics.

2.1 UOB Declarations

UOB's *(Units of Behaviour)* are IDEF3's most fundamental building blocks. They are identified with PSL's activities. Furthermore, IDEF3 distinguishes UOB's (aka *generic activities*), UOB-uses which are the occurences of UOB's in particular IDEF3 schematics, and UOB *activations* which are collections of instances of UOB-uses that satisfy the temporal and logical constraints imposed by an IDEF3 schematic.

A UOB declaration has the generic form:

```
<UOB-declaration> ::=
(define-UOB <con>
    [:documentation <string>]
    [:objects (<con>+)]
    [:constraints (<sentence>+)])
```

A UOB declaration enables the specification of common properties of generic activities outside the scope of any particular process.

The `:objects` and `:constraints` slots define object types that must be present and constraints that must be met by any subclass of the UOB. Specific subclasses can then be created within the context of a particular process by UOB-use declaration, importing the generic UOB information via the `:use-of` slot in the UOB-use frame.

2.2 UOB-use Declarations

```
<UOB-use-declaration> ::=
(define-UOB-use <con>
    [:documentation <string>]
    [:use-of (<con>+)]
    :in-process <con>
    [:decompositions (<con>+)]
    [:constraints (<sentence>+)])
```

UOB-use declarations specify the general constraints that must be satisfied by activities that occur in a particular IDEF3 process specification. They can inherit objects and constraints via the `:use-of` mechanism from several generic UOB declarations. Furthermore if the UOB represented by a UOB-use in a particular schematic is highly complex, it may be *decomposed* into schematics of finer granularity by the use of the `:decompositions` slot. An UOB-use may have several different such decompositions, providing the ability to model the same process from various perspectives.

2.3 Process Declarations

```
<process-declaration> ::=
(define-process <con>
    [:documentation <string>]
    :components (<con>+)
    [:constraints (<sentence>+)])
```

An IDEF3 process declaration corresponds to a graphical IDEF3 process schematic. The `:components` slot is used to specify the UOB's and junctions that appear in the schematic.

2.4 Links

Links are used in IDEF3 schematics mainly to specify temporal constraints among the UOB's of a process schematic. By using additional constraints links can also be used to express logical, causal, natural and conventional relations[3]. There are three generic categories of links, that are:

- simple precedence links
- constrained precedence links
- relational links

One important thing to note is that IDEF3 links are connecting just UOB's. Although sharing the same graphical representation, the arrows connecting junctions and UOB's or junctions and junctions have different semantics, and are handled in the context of the junction they are connecting.

2.4.1 Simple Precedence Links

```
<simple-link-declaration> ::=
(define-link <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ <con>)
```

The semantics for precedence links schematics can be expressed in terms of possible activations of those schematics[3]. The simple precedence link expresses the constraint that any activation of the schematic in which it appears contains activations of its `:pred` and `:succ` slots and the activation of the `:succ` slot does not begin before the end of the activation of the `:pred` slot.

2.4.2 Left to Right Precedence Links

```
<left-to-right-link-declaration> ::=
(define-lr-link <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ <con>)
```

A left to right precedence link, apart from the constraint imposed by a simple precedence link imposes the constraint that any activation of the `:pred` slot must be followed by an activation of the `:succ` slot.

2.4.3 Right to Left Precedence Links

```
<right-to-left-link-declaration> ::=
(define-rl-link <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ <con>)
```

A right to left precedence link, apart from the constraint imposed by a simple precedence link imposes the constraint that any activation of the `:succ` slot must be preceded by an activation of the `:pred` slot.

2.4.4 Bidirectional Precedence Links

```
<bidirectional-link-declaration> ::=
(define-bi-link <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ <con>)
```

Bidirectional precedence link impose the constraints of simple precedence link, together with left to right and right to left constraints.

2.4.5 General Constrained Precedence Links

```
<general-link-declaration> ::=
(define-general-link <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ <con>
    :constraints (<sentence>+))
```

General constrained precedence links impose the constraints of simple precedence link, together with the constraints specified in the `:constraints` slot.

2.4.6 Relational Links

```
<relational-link-declaration> ::=
(define-relational-link <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ <con>
    :constraints (<sentence>+))
```

Relational links have no predefined semantics. They just impose the constraints present in the `:constraints` slot.

2.5 Junctions

Brancing is represented in IDEF3 using junctions. A process can branch (converge or diverge) into multiple parallel (AND junction) or alternative (OR or XOR junctions) subprocesses. Also, branching can be done in asynchronuous (default) or synchronuous mode. Particular junctions exist for all those combinations, and their exact semantics are defined in [3]. We will just elaborate here on the syntax used for the various junction declarations, reffering the reader to [3] for the exact semantics.

2.5.1 Fan Out AND Junctions Declaration

```
<fan-out-and-junction-declaration> ::=
(define-fan-out-&-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ (<con>+)
    [:constraints (<sentence>+)])
```

2.5.2 Fan Out OR Junctions Declaration

```
<fan-out-or-junction-declaration> ::=
(define-fan-out-O-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ (<con>+)
    [:constraints (<sentence>+)])
```

2.5.3 Fan Out XOR Junctions Declaration

```
<fan-out-xor-junction-declaration> ::=
(define-fan-out-X-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ (<con>+)
    [:constraints (<sentence>+)])
```

2.5.4 Fan In AND Junctions Declaration

```
<fan-in-and-junction-declaration> ::=
(define-fan-in-&-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred (<con>+)
    :succ <con>
    [:constraints (<sentence>+)])
```

2.5.5 Fan In OR Junctions Declaration

```
<fan-in-or-junction-declaration> ::=
(define-fan-in-O-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred (<con>+)
    :succ <con>
    [:constraints (<sentence>+)])
```

2.5.6 Fan In XOR Junctions Declaration

```
<fan-in-xor-junction-declaration> ::=
(define-fan-in-X-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred (<con>+)
    :succ <con>
    [:constraints (<sentence>+)])
```

2.5.7 Synchronuous Fan Out AND Junctions Declaration

```
<sync-fan-out-and-junction-declaration> ::=
(define-sync-fan-out-&-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
```

```
    :succ (<con>+)
    [:constraints (<sentence>+)])
```

2.5.8 Synchronuous Fan Out OR Junctions Declaration

```
<sync-fan-out-or-junction-declaration> ::=
(define-sync-fan-out-O-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ (<con>+)
    [:constraints (<sentence>+)])
```

2.5.9 Synchronuous Fan Out XOR Junctions Declaration

```
<sync-fan-out-xor-junction-declaration> ::=
(define-sync-fan-out-X-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred <con>
    :succ (<con>+)
    [:constraints (<sentence>+)])
```

2.5.10 Synchronuous Fan In AND Junctions Declaration

```
<sync-fan-in-and-junction-declaration> ::=
(define-sync-fan-in-&-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred (<con>+)
    :succ <con>
    [:constraints (<sentence>+)])
```

2.5.11 Synchronuous Fan In OR Junctions Declaration

```
<sync-fan-in-or-junction-declaration> ::=
(define-sync-fan-in-O-junction <con>
    [:documentation <string>]
    :in-process <con>
    :pred (<con>+)
    :succ <con>
    [:constraints (<sentence>+)])
```

2.5.12 Synchronuous Fan In XOR Junctions Declaration

```
<sync-fan-in-xor-junction-declaration> ::=
(define-sync-fan-in-X-junction <con>
```

```
[:documentation <string>]
:in-process <con>
:pred (<con>+)
:succ <con>
[:constraints (<sentence>+)])
```

3 PSL extensions

In order to be able to capture the semantic concepts of IDEF3, some extensions were created for the PSL language.

Those extensions fall into several broad categories, that deal with *splitting* of processes, type-instance relationships for both activities and objects, temporal sequencing of activities, etc.

3.1 PSL Splitting extensions

Three new predicates `or_split`, `and_split` and `xor_split` are introduced. They have two (type level) activity arguments, the intended semantics being that the first one is a complex activity consisting of sub-activities whose activations are governed by the type of the split, appearing in the context of the second argument.

They are introduced by the following axioms:

3.1.1 OR Splits

Axiom 1.

```
(defrelation or_split (?a ?c) :=

(=> (or_split ?a ?c)
    (and (Activity ?a)
         (Activity ?c)))
```

`or_split` holds between activities.

Axiom 2.

```
(=> (or_split ?a ?c)
    (subactivity ?a ?c))
```

The complex activity occurs in the context of (as a subactivity of) the process.

Axiom 3.

```
(=> (and (or_split ?a ?c)
         (exists (?aa : (activation-of ?aa ?a))))
     (exists (?a1 : (subactivity ?a1 ?a))
             (exists ?aa1 (activation-of ?aa1 ?a1))))
)
```

If there exists an activation of the complex activity, there must exist an activation of al least one of its sub-activities.

3.1.2 AND Splits

Axiom 4.

```
(defrelation and_split (?a ?c) :=

(=> (and_split ?a ?c)
    (and (Activity ?a)
         (Activity ?c)))
```

`and_split` holds between activities.

Axiom 5.

```
(=> (and_split ?a ?c)
    (subactivity ?a ?c))
```

The complex activity occurs in the context of (as a subactivity of) the process.

Axiom 6.

```
(=> (and (and_split ?a ?c)
         (exists (?aa : (activation-of ?aa ?a))))
     (forall (?a1 : (subactivity ?a1 ?a))
             (exists ?aa1 (activation-of ?aa1 ?a1))))
)
```

If there exists an activation of the complex activity, there must exist activations of all of its sub-activities.

3.1.3 XOR Splits

Axiom 7.

```
(defrelation xor_split (?a ?c) :=

(=> (xor_split ?a ?c)
    (and (Activity ?a)
         (Activity ?c)))
```

`xor_split` holds between activities.

Axiom 8.

```
(=> (xor_split ?a ?c)
    (subactivity ?a ?c))
```

The complex activity occurs in the context of (as a subactivity of) the process.

Axiom 9.

```
(=> (and (xor_split ?a ?c)
          (exists (?aa : (activation-of ?aa ?a))))
     (exists-1 (?a1 : (subactivity ?a1 ?a))
               (exists ?aa1 (activation-of ?aa1 ?a1))))
)
```

If there exists an activation of the complex activity, there must exist an activation of exactly one of its sub-activities.

3.2 Sincronizing Axioms

We intoroduce two predicates, `sync_start` and `sync_finish` that hold between a complex activity and a process if all activations of the complex activity's subactivities start, respectively finish in a synchronized manner.

The two predicates are introduced by the following axioms:

Axiom 10.

```
(defrelation sync_start (?a ?c) :=

(=> (sync_start ?a ?c)
    (and (Activity ?a)
         (Activity ?c)))
```

`sync_start` holds between activities.

Axiom 11.

```
(=> (sync_start ?a ?c)
    (subactivity ?a ?c))
```

The complex activity occurs in the context of (as a subactivity of) the process.

Axiom 12.

```
(=> (sync_start ?a ?c)
    (forall (?a1 ?a2 : (and (activation-of ?a1 ?A1)
                            (activation-of ?a2 ?A2)
                            (subactivity ?A1 ?a)
                            (subactivity ?A2 ?a)))
                 (= (BeginOf ?a1)
                    (BeginOf ?a2))))
)
```

All activations of the complex activity's subactivities start in a synchronized manner.

Axiom 13.

```
(defrelation sync_finish (?a ?c) :=

(=> (sync_finish ?a ?c)
    (and (Activity ?a)
         (Activity ?c)))
```

sync_finish holds between activities.

Axiom 14.

```
(=> (sync_finish ?a ?c)
    (subactivity ?a ?c))
```

The complex activity occurs in the context of (as a subactivity of) the process.

Axiom 15.

```
(=> (sync_finish ?a ?c)
    (forall (?a1 ?a2 : (and (activation-of ?a1 ?A1)
                            (activation-of ?a2 ?A2)
                            (subactivity ?A1 ?a)
                            (subactivity ?A2 ?a)))
                 (= (EndOf ?a1)
                    (EndOf ?a2))))
)
```

All activations of the complex activity's subactivities finish in a synchronized manner.

3.3 Syncronuos Splits

All splits have both syncronuos start and syncronuos finish versions, which are introduced by inheriting both the split's properties and those that define the respective syncronicity.

Axiom 16.

```
(defrelation sync_start_and_split (?a ?c) :=

(<=> (sync_start_and_split ?a ?c)
     (and (sync_start ?a ?c)
          (and_split ?a ?c)))
)
```

Axiom 17.

```
(defrelation sync_start_or_split (?a ?c) :=

(<=> (sync_start_or_split ?a ?c)
     (and (sync_start ?a ?c)
          (or_split ?a ?c)))
)
```

Axiom 18.

```
(defrelation sync_start_xor_split (?a ?c) :=

(<=> (sync_start_xor_split ?a ?c)
     (and (sync_start ?a ?c)
          (xor_split ?a ?c)))
)
```

Axiom 19.

```
(defrelation sync_finish_and_split (?a ?c) :=

(<=> (sync_finish_and_split ?a ?c)
     (and (sync_finish ?a ?c)
          (and_split ?a ?c)))
)
```

Axiom 20.

```
(defrelation sync_finish_or_split (?a ?c) :=

(<=> (sync_finish_or_split ?a ?c)
```

```
     (and (sync_finish ?a ?c)
          (or_split ?a ?c)))
)
```

Axiom 21.

```
(defrelation sync_finish_xor_split (?a ?c) :=

(<=> (sync_finish_xor_split ?a ?c)
     (and (sync_finish ?a ?c)
          (xor_split ?a ?c)))
)
```

3.4 Temporal Sequencing of Activities

A new predicate, `follows` is introduced with the intended semantics for `(follows U J P)` being that "activations of `J` come after activations of `U` inside process `P`, in an *IDEF3 link sense*."

That *IDEF3 link sense* is captured in the following axiom:

Axiom 22.

```
(defrelation follows (?A ?B ?P) :=

(=> (follows ?A ?B ?P)
    (forall (?a : (activation-of ?a ?A))
      (forall (?p : (activation-of ?p ?P))
        (=> (subactivity-occurence  ?a ?p)
            (exists (?b : (activation-of ?b ?B))
               (and (subactivity-occurence ?b ?p)
                    (BeforeEq (EndOf ?a)
                              (BeginOf ?b)))))))))
```

3.5 IDEF3 Processes

An interesting issue that occured when doing the IDEF3 to PSL translation was *What is the right abstraction level in PSL at which to do the translation?*

As far as theIDEF3 to PSL translator is concerned, the level of abstraction at which the translation is done doesn't really matter. One has to think of translation rules that go all the way to the bottom (i.e. the PSL primitive concepts) and define the IDEF3 constructs in terms of these. Alternatively, once semantic agreement has been checked by going to the lowest level, the compilation rules can be written in terms of higher level concepts reused from the PSL Ontology, or in terms of concepts defined in a PSL extension.

However, to facilitate the style currently chosen for translating out of PSL the higher level approach was chosen. That is new high level semantic concepts were defined in terms of the PSL Ontology and the compilation rules were written in terms of those concepts.

So, a new predicate `idef-process` that characterizes IDEF3 processes was introduced as an extension. Its definition is:

```
(defrelation idef-process (?P) :=

(=> (idef-process ?P)
    (forall (?p : (activation-of ?p ?P))
        (exists (?u ?U : (subactivity-occurence ?u ?p)
                         (activation-of ?u ?U)
                         (subactivity ?U ?P))
           (and (forall (?V : (subactivity ?V ?P))
                    (not (follows ?V ?U ?P)))
                (= (beginof ?u) (beginof ?p)))))))
```

3.6 Type-Instance Relationships

Both activities and objects are type-level in IDEF3 and in PSL. However, they are encoded as predicates in IDEF3, but as objects in PSL. So, going with the PSL encoding we choose to reify both activity and object types, introducing two new predicates to express their type-instance relationship.

We choosed to reify both activity and object types, introducing two new predicates to express their type-instance relationship.

The two predicates are `(activation-of ?a ?A)` and `(instance-of ?o ?O)` both having the intended semantics that their first argument is an instance of the second. (for activities, respectively objects)

For the moment they are both defined in terms of the PSL `occurence` as:

```
(defrelation activation-of (?a ?A) :=

(<=> (activation-of ?a ?A)
     (occurence ?a ?A)))


(defrelation instance-of (?a ?A) :=

(<=> (instance-of ?a ?A)
     (occurence ?a ?A)))
```

but this might change in the future.

3.7 Documentation

A new predicate, `doc` was introduced, that holds between an object and its documentation.

4 Compilation rules

The PSL representation of an IDEF3 schematic will be a set of (implicitly and-ed) KIF sentences that define a PSL theory. The translation process can be described by a set of meta-theoretic *compilation rules* that associate KIF sentences with the IDEF3 constructs. (writing such compilation rules can be also seen as providing a formal, declarative semantics into PSL to IDEF3 constructs)

The notion of compilation will be defined relative to a process specification Π for each type of IDEF3 declaration and rules for the compilation of the individual slots of a declaration will be written in the style used in [2].

4.1 Compiling UOB Declarations

The compilation $Comp_\Pi(\mu)$ of a UOB declaration having the generic form:

(define-UOB γ
    :documentation δ
    :objects ($\xi_1 \ldots \xi_n$)
    :constraints ($\kappa_1 \ldots \kappa_m$))

is defined as (and $Comp_\Pi$(:documentation) $Comp_\Pi$(:objects) $Comp_\Pi$(:constraints)), where:

$Comp_\Pi$(:documentation) is the sentence:

(doc γ δ)

$Comp_\Pi$(:objects) is the sentence:

(forall (?a : (activation-of ?a γ))
  (exists ($\eta_1 \ldots \eta_n$: (instance-of η_1 ξ_1) ... (instance-of η_n ξ_n))
    (and (in η_1 ξ_1) ... (in η_n ξ_n))))

$Comp_\Pi$(:constraints) is the sentence:

(and $\kappa_1 \ldots \kappa_m$)

4.2 Compiling UOB-use Declarations

The compilation $Comp_\Pi(\mu)$ of a UOB-use declaration having the generic form:

(define-UOB-use α
    :documentation δ
    :use-of ($\gamma_1 \ldots \gamma_n$)
    :in-process π
    :decompositions ($\rho_1 \ldots \rho_k$)
    :constraints ($\kappa_1 \ldots \kappa_m$))

is defined as (and $Comp_\Pi$(:documentation) $Comp_\Pi$(:use-of) $Comp_\Pi$(:in-process) $Comp_\Pi$(:decompositions) $Comp_\Pi$(:constraints)), where:

$Comp_\Pi$(:documentation) is the sentence:

(doc α δ)

$Comp_\Pi$(:use-of) is the sentence:

```
(forall (?a : (activation-of ?a α))
  (and (activation-of ?a γ1) ... (activation-of ?a γn)))
```

$Comp_\Pi$(:in-process) is the sentence:

```
(forall (?a : (activation-of ?a α))
  (exists (?p : (activation-of ?p π))
    (subactivity-occurence ?a ?p)))
```

$Comp_\Pi$(:decompositions) is the sentence:

```
(forall (?a : (activation-of ?a α))
  (and (activation-of ?a ρ1) ... (activation-of ?a ρk)))
```

$Comp_\Pi$(:constraints) is the sentence:

(and $\kappa_1 \ldots \kappa_m$)

4.3 Compiling Process Declarations

The compilation $Comp_\Pi(\pi)$ of a process declaration having the generic form:

```
(define-process π
    :documentation δ
    :components (α1 ... αn)
    :constraints (κ1 ... κm))
```

is defined as (and $Comp_\Pi$(:documentation) $Comp_\Pi$(:components) $Comp_\Pi$(:constraints)), where:

$Comp_\Pi$(:documentation) is the sentence:

(doc π δ)

$Comp_\Pi$(:components) is the sentence:

```
(and (subactivity α1 π)
     (subactivity α2 π)
         ....
     (subactivity αn π)
     (idef-process π))
```

$Comp_\Pi$(:constraints) is the sentence:

(and $\kappa_1 \ldots \kappa_m$)

4.4 Compiling Simple Precedence Links Declarations

The compilation $Comp_{\Pi}(\lambda)$ of a simple precedence link declaration having the generic form:

```
(define-link λ
    :documentation δ
    :in-process π
    :pred α
    :succ β)
```

is defined as (and $Comp_{\Pi}$(:documentation) $Comp_{\Pi}$(:in-process:pred:succ)), where:

$Comp_{\Pi}$(:documentation) is the sentence:

```
(doc λ δ)
```

$Comp_{\Pi}$(:in-process:pred:succ) is the sentence:

```
(follows α β π)
```

4.5 Compiling Left to Right Precedence Links Declarations

The compilation $Comp_{\Pi}(\lambda)$ of a left to right precedence link declaration having the generic form:

```
(define-lr-link λ
    :documentation δ
    :in-process π
    :pred α
    :succ β)
```

is defined as (and $Comp_{\Pi}$(:documentation) $Comp_{\Pi}$(:in-process:pred:succ) $Comp_{\Pi}$(lr)), where:

$Comp_{\Pi}$(:documentation) is the sentence:

```
(doc λ δ)
```

$Comp_{\Pi}$(:in-process:pred:succ) is the sentence:

```
(follows α β π)
```

$Comp_{\Pi}$(lr) is the sentence:

```
(forall (?p : (activation-of ?p α))
  (exists (?s : (activation-of ?s β))
    (Before (Endof ?p) (Beginof ?s))))
```

4.6 Compiling Right to Left Precedence Links Declarations

The compilation $Comp_{\Pi}(\lambda)$ of a right to left precedence link declaration having the generic form:

```
(define-rl-link λ
    :documentation δ
    :in-process π
    :pred α
    :succ β)
```

is defined as `(and` $Comp_{\Pi}$`(:documentation)` $Comp_{\Pi}$`(:in-process:pred:succ)` $Comp_{\Pi}$`(rl))`, where:

$Comp_{\Pi}$`(:documentation)` is the sentence:

```
(doc λ δ)
```

$Comp_{\Pi}$`(:in-process:pred:succ)` is the sentence:

```
(follows α β π)
```

$Comp_{\Pi}$(rl) is the sentence:

```
(forall (?s : (activation-of ?s β))
  (exists (?p : (activation-of ?p α))
    (Before (Endof ?p) (Beginof ?s))))
```

4.7 Compiling Bidirectional Precedence Links Declarations

The compilation $Comp_{\Pi}(\lambda)$ of a bidirectional precedence link declaration having the generic form:

```
(define-bi-link λ
    :documentation δ
    :in-process π
    :pred α
    :succ β)
```

is defined as `(and` $Comp_{\Pi}$`(:documentation)` $Comp_{\Pi}$`(:in-process:pred:succ)` $Comp_{\Pi}$`(lr)` $Comp_{\Pi}$`(rl))`, where:

$Comp_{\Pi}$`(:documentation)` is the sentence:

```
(doc λ δ)
```

$Comp_{\Pi}$`(:in-process:pred:succ)` is the sentence:

```
(follows α β π)
```

$Comp_\Pi$(lr) is the sentence:

```
(forall (?p : (activation-of ?p α))
  (exists (?s : (activation-of ?s β))
    (Before (Endof ?p) (Beginof ?s))))
```

$Comp_\Pi$(rl) is the sentence:

```
(forall (?s : (activation-of ?s β))
  (exists (?p : (activation-of ?p α))
    (Before (Endof ?p) (Beginof ?s))))
```

4.8 Compiling General Constrained Precedence Links Declarations

The compilation $Comp_\Pi(\lambda)$ of a general constrained precedence link declaration having the generic form:

```
(define-link λ
    :documentation δ
    :in-process π
    :pred α
    :succ β)
    :constraints (κ1 ... κm))
```

is defined as `(and` $Comp_\Pi$`(:documentation)` $Comp_\Pi$`(:in-process:pred:succ)` $Comp_\Pi$`(:constraints))`, where:

$Comp_\Pi$`(:documentation)` is the sentence:

```
(doc λ δ)
```

$Comp_\Pi$`(:in-process:pred:succ)` is the sentence:

```
(follows α β π)
```

$Comp_\Pi$`(:constraints)` is the sentence:

```
(and κ1 ... κm)
```

4.9 Compiling Relational Links Declarations

The compilation $Comp_\Pi(\lambda)$ of a relational link declaration having the generic form:

```
(define-link λ
    :documentation δ
    :in-process π
    :pred α
    :succ β)
    :constraints (κ1 ... κm))
```

is defined as (and $Comp_\Pi$(:documentation) $Comp_\Pi$(:constraints)), where:

$Comp_\Pi$(:documentation) is the sentence:

```
(doc λ δ)
```

$Comp_\Pi$(:constraints) is the sentence:

```
(and κ₁ ... κₘ)
```

4.10 Compiling Fan Out Junction Declarations

Fan Out junctions are identified with complex activities, having as sub-activities the activities denoted by the UOB's following the junction. They are compiled using the appropriate PSL split extension, which are defined in Section 3.

Since there are many kinds of fan-out junctions in the IDEF3 language, we will give a generic compilation schema that applies to all of them.

The compilation $Comp_\Pi(\zeta)$ of a fan-out junction declaration having the generic form:

```
(define[-sync]-fan-out-σ-junction ζ
    :documentation δ
    :in-process π
    :pred α
    :succ (β₁ ... βₙ)
    :constraints (κ₁ ... κₘ))
```

is defined as (and $Comp_\Pi$(:documentation) $Comp_\Pi$(:in-process) $Comp_\Pi$(:pred) $Comp_\Pi$(:succ) $Comp_\Pi$(:constraints)), where:

$Comp_\Pi$(:documentation) is the sentence:

```
(doc ζ δ)
```

$Comp_\Pi$(:in-process) is the sentence:

```
(forall (?j : (activation-of ?j ζ))
  (exists (?p : (activation-of ?p π))
    (subactivity-occurence ?j ?p)))
```

$Comp_\Pi$(:pred) is the sentence:

```
(follows α ζ π)
```

$Comp_\Pi$(:succ) is the sentence:

```
(and (ψ ζ π)
     (subactivity β₁ ζ)
     ...
     (subactivity βₙ ζ))
```

where ψ is one of `or_split`, `and_split` or `xor_split` for σ standing for `O`, `&` respectively `X` or one of `sync_start_or_split`, `sync_start_and_split` or `sync_start_xor_split` for the respective syncronuos junctions.

Again this is just a shorthand for writing the compilation rules for all fan-out junctions in a single compilation schema.

$Comp_\Pi$(`:constraints`) is the sentence:

(and $\kappa_1 \ldots \kappa_m$)

4.11 Compiling Fan In Junction Declarations

Fan In junctions are identified with complex activities, having as sub-activities the activities denoted by the UOB's preceding the junction. They are compiled using the appropriate PSL split extension, which are defined in Section 3.

Since there are many kinds of fan-in junctions in the IDEF3 language, we will give a generic compilation schema that applies to all of them.

The compilation $Comp_\Pi(\zeta)$ of a fan-in junction declaration having the generic form:

```
(define[-sync]-fan-in-σ-junction ζ
     :documentation δ
     :in-process π
     :pred (α1 ... αn)
     :succ β
     :constraints (κ1 ... κm))
```

is defined as (and $Comp_\Pi$(`:documentation`) $Comp_\Pi$(`:in-process`) $Comp_\Pi$(`:pred`) $Comp_\Pi$(`:succ`) $Comp_\Pi$(`:constraints`)), where:

$Comp_\Pi$(`:documentation`) is the sentence:

(doc ζ δ)

$Comp_\Pi$(`:in-process`) is the sentence:

```
(forall (?j : (activation-of ?j ζ))
  (exists (?p : (activation-of ?p π))
    (subactivity-occurence ?j ?p)))
```

$Comp_\Pi$(`:pred`) is the sentence:

```
(and (ψ ζ π)
      (subactivity α1 ζ)
      ...
      (subactivity αn ζ))
```

where ψ is one of `or_split`, `and_split` or `xor_split` for σ standing for `O`, `&` respectively `X` or one of `sync_finish_or_split`, `sync_finish_and_split` or `sync_finish_xor_split` for the respective syncronuos junctions.

Again this is just a shorthand for writing the compilation rules for all fan-in junctions in a single compilation schema.

$Comp_{\Pi}$(:succ) is the sentence:

(follows ζ β π)

$Comp_{\Pi}$(:constraints) is the sentence:

(and $\kappa_1 \ldots \kappa_m$)

5 The translator

Once the compilation rules were written, implementing the translator was a trivial task. The compilation rules were written as `lisp` macros, and the translator itself just calls `macroexpand-1` for all the forms of the IDEF3 file and stores the results in the PSL file.

The translator was later provided with a KIF expression simplifier which can be used to convert from the higher level KIF expressions generated by the compilation rules, that may contain a form of typed quantifiers and polymorphic operators to a simple syntax form that uses typeless quantifiers and standard operators. A kif-to-prolog extension that was needed for subsequent translation steps was also included, so the translator can now generate three output formats for an IDEF3 file, that is PSL terminology using KIF syntax, PSL terminology using the simplified KIF syntax, and PSL terminology using Prolog syntax.

The translator has been dumped to an executable image which is invoked as:

i2p k|s|p *ideffile pslfile*

where k|s|p selects one of the syntax options described above.

A The Camile scenario

A.1 IDEF3 source

```
;;
;; Camile Scenario IDEF 3 representation
;;
;; Mihai Ciocoiu
;;

(define-UOB Make-GT350
    :documentation "Make GT350")

(define-UOB Make-Interior
    :documentation "Make Interior"
    :objects (A002-bench))

(define-UOB Make-Drive
    :documentation "Make Drive"
    :objects (A003-bench))

(define-UOB Make-Trim
    :documentation "Make Trim"
    :objects (A002-bench A005-bench))

(define-UOB Make-Chassis
    :documentation "Make Chassis")

(define-UOB Final-Assembly
    :documentation "Final Assembly")

(define-UOB Make-Block
    :documentation "Make Block")

(define-UOB Make-Hamess
    :documentation "Make Hamess")

(define-UOB Make-Wires
    :documentation "Make Wires")

(define-UOB Assemble-Engine
    :documentation "Assemble Engine")

(define-UOB Produce-Moulded-Metal-Item
    :documentation "Produce Moulded Metal Item")

(define-UOB Machine-Block
    :documentation "Machine Block")

(define-UOB Change-Mould
    :documentation "Change Mould")

(define-UOB Setup-Furnace
```

```
    :documentation "Setup Furnace")

(define-UOB Analyse-Metal
    :documentation "Analyse Metal")

(define-UOB Melt
    :documentation "Melt")

(define-UOB Mould
    :documentation "Mould")

(define-UOB Wait
    :documentation "Wait")

(define-UOB Clear-Cavities
    :documentation "Clear Cavities")

(define-UOB Setup-Racks
    :documentation "Setup Racks")

(define-UOB Pour
    :documentation "Pour")

(define-UOB Remove-Racks
    :documentation "Remove Racks")

(define-UOB Batch-Complete
    :documentation "Batch Complete")



(define-process Make-GT350-proc
    :documentation "The Top Level GT350 process schematic"
    :components (Make-GT350-1)
    :constraints nil)

(define-UOB-use Make-GT-350-1
    :documentation "The occurence of Make-GT350 in the schematic"
    :use-of (Make-GT350)
    :decompositions (Decomposition-19)
    :constraints (duration make_engine-1 5))



(define-process Decomposition-19
    :documentation "Decomposition of Make GT350"
    :components (Make-Interior-1 Make-Drive-1 Make-Trim-1
 Make-Engine-1 Make-Chassis-1
 Final-Assembly-1 J2 J1)
```

```
    :constraints nil)

(define-UOB-use Make-Interior-1
    :documentation "The occurence of Make-Interior in the Dec-19 schematic"
    :in-process Decomposition-19
    :use-of (Make-Interior)
    :decompositions nil
    :constraints nil)

(define-UOB-use Make-Drive-1
    :documentation "The occurence of Make-Drive in the Dec-19 schematic"
    :in-process Decomposition-19
    :use-of (Make-Drive)
    :decompositions nil
    :constraints nil)

(define-UOB-use Make-Trim-1
    :documentation "The occurence of Make-Trim in the Dec-19 schematic"
    :in-process Decomposition-19
    :use-of (Make-Trim)
    :decompositions nil
    :constraints nil)

(define-UOB-use Make-Engine-1
    :documentation "The occurence of Make-Engine in the Dec-19 schematic"
    :in-process Decomposition-19
    :use-of (Make-Engine)
    :decompositions (Decomposition-26)
    :constraints nil)

(define-UOB-use Make-Chassis-1
    :documentation "The occurence of Make-Chassis in the Dec-19 schematic"
    :in-process Decomposition-19
    :use-of (Make-Chassis)
    :decompositions nil
    :constraints nil)

(define-UOB-use Final-Assembly-1
    :documentation "The occurence of Final-Assembly in the Dec-19 schematic"
    :in-process Decomposition-19
    :use-of (Final-Assembly)
    :decompositions nil
    :constraints nil)

(define-fan-in-&-junction J2
    :documentation "J2"
    :in-process Decomposition-19
    :pred (Make-Interior-1 Make-Drive-1 Make-Trim-1 Make-Engine-1 Make-Chassis-1)
    :succ Final-Assembly-1
    :constraints nil)
```

```
(define-fan-out-&-junction J1
    :documentation "J1"
    :in-process Decomposition-19
    :pred nil
    :succ (Make-Interior-1 Make-Drive-1 Make-Trim-1 Make-Engine-1 Make-Chassis-1)
    :constraints nil)



(define-process Decomposition-26
    :documentation "Decomposition of Make Engine"
    :components (Make-Block-1 Make-Harness-1 Make-Wires-1 Assemble-Engine-1 J4 J3)
    :constraints nil)

(define-UOB-use Make-Block-1
    :documentation "The occurence of Make-Block in the Dec-26 schematic"
    :in-process Decomposition-26
    :use-of (Make-Block)
    :decompositions (Decomposition-27)
    :constraints nil)

(define-UOB-use Make-Harness-1
    :documentation "The occurence of Make-Harness in the Dec-26 schematic"
    :in-process Decomposition-26
    :use-of (Make-Harness)
    :decompositions nil
    :constraints nil)

(define-UOB-use Make-Wires-1
    :documentation "The occurence of Make-Wires in the Dec-26 schematic"
    :in-process Decomposition-26
    :use-of (Make-Wires)
    :decompositions nil
    :constraints nil)

(define-UOB-use Assemble-Engine-1
    :documentation "The occurence of Assemble-Engine in the Dec-26 schematic"
    :in-process Decomposition-26
    :use-of (Assemble-Engine)
    :decompositions nil
    :constraints nil)

(define-fan-in-&-junction J4
    :documentation "J4"
    :in-process Decomposition-26
    :pred (Make-Block-1 Make-Harness-1 Make-Wires-1)
    :succ Assemble-Engine-1
    :constraints nil)
```

```
(define-fan-out-&-junction J3
    :documentation "J3"
    :in-process Decomposition-26
    :pred nil
    :succ (Make-Block-1 Make-Harness-1 Make-Wires-1)
    :constraints nil)



(define-process Decomposition-27
    :documentation "Decomposition of Make Block"
    :components (Produce-Moulded-Metal-Item Machine-Block)
    :constraints nil)

(define-UOB-use Produce-Moulded-Metal-Item-1
    :documentation "The occurence of Produce-Moulded-Metal-Item in the Dec-27 schematic"
    :in-process Decomposition-27
    :use-of (Produce-Moluded-Metal-Item)
    :decompositions (Decomposition-1)
    :constraints nil)

(define-UOB-use Machine-Block-1
    :documentation "The occurence of Machine-Block in the Dec-27 schematic"
    :in-process Decomposition-27
    :use-of (Machine-Block)
    :decompositions nil
    :constraints nil)

(define-link L93
    :documentation "L93"
    :in-process Decomposition-27
    :pred Produce-Moluded-Metal-Item-1
    :succ Machine-Block-1
    :constraints nil)



(define-process Decomposition-1
    :documentation "Decomposition of Produce Moulded Metal Item"
    :components (Change-Mould-1 Setup-Furnace-1 Analyse-Metal-1 Melt-1
Mould-1 Wait-1 Clear-Cavities-1 J8 J7 J6 J5)
    :constraints nil)

(define-UOB-use Change-Mould-1
    :documentation "The occurence of Change-Mould in the Dec-1 schematic"
    :in-process Decomposition-1
    :use-of (Change-Mould)
    :decompositions nil
    :constraints nil)
```

```
(define-UOB-use Setup-Furnace-1
    :documentation "The occurence of Setup-Furnace in the Dec-1 schematic"
    :in-process Decomposition-1
    :use-of (Setup-Furnace)
    :decompositions nil
    :constraints nil)

(define-UOB-use Analyse-Metal-1
    :documentation "The occurence of Analyse-Metal in the Dec-1 schematic"
    :in-process Decomposition-1
    :use-of (Analyse-Metal)
    :decompositions nil
    :constraints nil)

(define-UOB-use Melt-1
    :documentation "The occurence of Melt in the Dec-1 schematic"
    :in-process Decomposition-1
    :use-of (Melt)
    :decompositions nil
    :constraints nil)

(define-UOB-use Mould-1
    :documentation "The occurence of Mould in the Dec-1 schematic"
    :in-process Decomposition-1
    :use-of (Mould)
    :decompositions (Decomposition-1-1)
    :constraints nil)

(define-UOB-use Wait-1
    :documentation "The occurence of Wait in the Dec-1 schematic"
    :in-process Decomposition-1
    :use-of (Wait)
    :decompositions nil
    :constraints nil)

(define-UOB-use Clear-Cavities-1
    :documentation "The occurence of Clear-Cavities in the Dec-1 schematic"
    :in-process Decomposition-1
    :use-of (Clear-Cavities)
    :decompositions nil
    :constraints nil)

(define-fan-in-&-junction J8
    :documentation "J8"
    :in-process Decomposition-1
    :pred (Analyse-Metal-1 Melt-1)
    :succ Mould-1
    :constraints nil)

(define-fan-out-&-junction J7
```

```
    :documentation "J7"
    :in-process Decomposition-1
    :pred Setup-Furnace-1
    :succ (Analyse-Metal-1 Melt-1)
    :constraints nil)

(define-fan-in-X-junction J6
    :documentation "J6"
    :in-process Decomposition-1
    :pred (J5 Change-Mould-1)
    :succ Setup-Furnace-1
    :constraints nil)

(define-fan-out-&-junction J5
    :documentation "J5"
    :in-process Decomposition-1
    :pred nil
    :succ (J6 Change-Mould-1)
    :constraints nil)

(define-link L120
    :documentation "L120"
    :in-process Decomposition-1
    :pred Mould-1
    :succ Wait-1
    :constraints nil)

(define-link L123
    :documentation "L123"
    :in-process Decomposition-1
    :pred Wait-1
    :succ Clear-Cavities-1
    :constraints nil)



(define-process Decomposition-1-1
    :documentation "Decomposition of Mould"
    :components (Setup-Racks-1 Pour-1 Remove-Racks-1 Batch-Complete-1)
    :constraints nil)

(define-UOB-use Setup-Racks-1
    :documentation "The occurence of Setup-Racks in the Dec-1-1 schematic"
    :in-process Decomposition-1-1
    :use-of (Setup-Racks)
    :decompositions nil
    :constraints nil)

(define-UOB-use Pour-1
    :documentation "The occurence of Pour in the Dec-1-1 schematic"
```

```
    :in-process Decomposition-1-1
    :use-of (Pour)
    :decompositions nil
    :constraints nil)

(define-UOB-use Remove-Racks-1
    :documentation "The occurence of Remove-Racks in the Dec-1-1 schematic"
    :in-process Decomposition-1-1
    :use-of (Remove-Racks)
    :decompositions nil
    :constraints nil)

(define-UOB-use Batch-Complete-1
    :documentation "The occurence of Batch-Complete in the Dec-1-1 schematic"
    :in-process Decomposition-1-1
    :use-of (Batch-Complete)
    :decompositions nil
    :constraints nil)
```

A.2 PSL KIF representation

```
(doc make-gt350 "Make GT350")


(and (doc make-interior "Make Interior")
     (forall (?a : (activation-of ?a make-interior))
      (exists (?o1 : (instance-of ?o1 a002-bench)) (in ?o1 ?a))))


(and (doc make-drive "Make Drive")
     (forall (?a : (activation-of ?a make-drive))
      (exists (?o1 : (instance-of ?o1 a003-bench)) (in ?o1 ?a))))


(and (doc make-trim "Make Trim")
     (forall (?a : (activation-of ?a make-trim))
      (exists
       (?o1 ?o2 : (instance-of ?o1 a002-bench)
        (instance-of ?o2 a005-bench))
       (and (in ?o1 ?a) (in ?o2 ?a)))))


(doc make-chassis "Make Chassis")


(doc final-assembly "Final Assembly")


(doc make-block "Make Block")


(doc make-hamess "Make Hamess")


(doc make-wires "Make Wires")


(doc assemble-engine "Assemble Engine")


(doc produce-moulded-metal-item "Produce Moulded Metal Item")


(doc machine-block "Machine Block")


(doc change-mould "Change Mould")


(doc setup-furnace "Setup Furnace")


(doc analyse-metal "Analyse Metal")


(doc melt "Melt")
```

```
(doc mould "Mould")


(doc wait "Wait")


(doc clear-cavities "Clear Cavities")


(doc setup-racks "Setup Racks")


(doc pour "Pour")


(doc remove-racks "Remove Racks")


(doc batch-complete "Batch Complete")


(and (doc make-gt350-proc "The Top Level GT350 process schematic")
     (subactivity make-gt350-1 make-gt350-proc)
     (idef-process make-gt350-proc))


(and (doc make-gt-350-1 "The occurence of Make-GT350 in the schematic")
     (forall (?a : (activation-of ?a make-gt-350-1))
      (activation-of ?a make-gt350))
     (forall (?a : (activation-of ?a make-gt-350-1))
      (activation-of ?a decomposition-19))
     (duration make_engine-1 5))


(and (doc decomposition-19 "Decomposition of Make GT350")
     (subactivity make-interior-1 decomposition-19)
     (subactivity make-drive-1 decomposition-19)
     (subactivity make-trim-1 decomposition-19)
     (subactivity make-engine-1 decomposition-19)
     (subactivity make-chassis-1 decomposition-19)
     (subactivity final-assembly-1 decomposition-19)
     (subactivity j2 decomposition-19)
     (subactivity j1 decomposition-19) (idef-process decomposition-19))


(and (doc make-interior-1
      "The occurence of Make-Interior in the Dec-19 schematic")
     (forall (?a : (activation-of ?a make-interior-1))
      (activation-of ?a make-interior))
     (forall (?a : (activation-of ?a make-interior-1))
      (exists (?p : (activation-of ?p decomposition-19))
       (subactivity-occurrence ?a ?p))))


(and (doc make-drive-1
```

```
      "The occurence of Make-Drive in the Dec-19 schematic")
     (forall (?a : (activation-of ?a make-drive-1))
      (activation-of ?a make-drive))
     (forall (?a : (activation-of ?a make-drive-1))
      (exists (?p : (activation-of ?p decomposition-19))
       (subactivity-occurrence ?a ?p))))


(and (doc make-trim-1
      "The occurence of Make-Trim in the Dec-19 schematic")
     (forall (?a : (activation-of ?a make-trim-1))
      (activation-of ?a make-trim))
     (forall (?a : (activation-of ?a make-trim-1))
      (exists (?p : (activation-of ?p decomposition-19))
       (subactivity-occurrence ?a ?p))))


(and (doc make-engine-1
      "The occurence of Make-Engine in the Dec-19 schematic")
     (forall (?a : (activation-of ?a make-engine-1))
      (activation-of ?a make-engine))
     (forall (?a : (activation-of ?a make-engine-1))
      (exists (?p : (activation-of ?p decomposition-19))
       (subactivity-occurrence ?a ?p)))
     (forall (?a : (activation-of ?a make-engine-1))
      (activation-of ?a decomposition-26)))


(and (doc make-chassis-1
      "The occurence of Make-Chassis in the Dec-19 schematic")
     (forall (?a : (activation-of ?a make-chassis-1))
      (activation-of ?a make-chassis))
     (forall (?a : (activation-of ?a make-chassis-1))
      (exists (?p : (activation-of ?p decomposition-19))
       (subactivity-occurrence ?a ?p))))


(and (doc final-assembly-1
      "The occurence of Final-Assembly in the Dec-19 schematic")
     (forall (?a : (activation-of ?a final-assembly-1))
      (activation-of ?a final-assembly))
     (forall (?a : (activation-of ?a final-assembly-1))
      (exists (?p : (activation-of ?p decomposition-19))
       (subactivity-occurrence ?a ?p))))


(and (doc j2 "J2")
     (forall (?j : (activation-of ?j j2))
      (exists (?p : (activation-of ?p decomposition-19))
       (subactivity-occurrence ?j ?p)))
     (follows j2 final-assembly-1 decomposition-19)
     (and (and_split j2 decomposition-19)
          (subactivity make-interior-1 j2)
          (subactivity make-drive-1 j2) (subactivity make-trim-1 j2)
          (subactivity make-engine-1 j2)
          (subactivity make-chassis-1 j2)))
```

```
(and (doc j1 "J1")
     (forall (?j : (activation-of ?j j1))
      (exists (?p : (activation-of ?p decomposition-19))
       (subactivity-occurrence ?j ?p)))
     (and (and_split j1 decomposition-19)
          (subactivity make-interior-1 j1)
          (subactivity make-drive-1 j1) (subactivity make-trim-1 j1)
          (subactivity make-engine-1 j1)
          (subactivity make-chassis-1 j1)))


(and (doc decomposition-26 "Decomposition of Make Engine")
     (subactivity make-block-1 decomposition-26)
     (subactivity make-harness-1 decomposition-26)
     (subactivity make-wires-1 decomposition-26)
     (subactivity assemble-engine-1 decomposition-26)
     (subactivity j4 decomposition-26)
     (subactivity j3 decomposition-26) (idef-process decomposition-26))


(and (doc make-block-1
      "The occurence of Make-Block in the Dec-26 schematic")
     (forall (?a : (activation-of ?a make-block-1))
      (activation-of ?a make-block))
     (forall (?a : (activation-of ?a make-block-1))
      (exists (?p : (activation-of ?p decomposition-26))
       (subactivity-occurrence ?a ?p)))
     (forall (?a : (activation-of ?a make-block-1))
      (activation-of ?a decomposition-27)))


(and (doc make-harness-1
      "The occurence of Make-Harness in the Dec-26 schematic")
     (forall (?a : (activation-of ?a make-harness-1))
      (activation-of ?a make-harness))
     (forall (?a : (activation-of ?a make-harness-1))
      (exists (?p : (activation-of ?p decomposition-26))
       (subactivity-occurrence ?a ?p))))


(and (doc make-wires-1
      "The occurence of Make-Wires in the Dec-26 schematic")
     (forall (?a : (activation-of ?a make-wires-1))
      (activation-of ?a make-wires))
     (forall (?a : (activation-of ?a make-wires-1))
      (exists (?p : (activation-of ?p decomposition-26))
       (subactivity-occurrence ?a ?p))))


(and (doc assemble-engine-1
      "The occurence of Assemble-Engine in the Dec-26 schematic")
     (forall (?a : (activation-of ?a assemble-engine-1))
      (activation-of ?a assemble-engine))
     (forall (?a : (activation-of ?a assemble-engine-1))
      (exists (?p : (activation-of ?p decomposition-26))
       (subactivity-occurrence ?a ?p))))
```

```
(and (doc j4 "J4")
     (forall (?j : (activation-of ?j j4))
      (exists (?p : (activation-of ?p decomposition-26))
       (subactivity-occurrence ?j ?p)))
     (follows j4 assemble-engine-1 decomposition-26)
     (and (and_split j4 decomposition-26) (subactivity make-block-1 j4)
          (subactivity make-harness-1 j4)
          (subactivity make-wires-1 j4)))


(and (doc j3 "J3")
     (forall (?j : (activation-of ?j j3))
      (exists (?p : (activation-of ?p decomposition-26))
       (subactivity-occurrence ?j ?p)))
     (and (and_split j3 decomposition-26) (subactivity make-block-1 j3)
          (subactivity make-harness-1 j3)
          (subactivity make-wires-1 j3)))


(and (doc decomposition-27 "Decomposition of Make Block")
     (subactivity produce-moulded-metal-item decomposition-27)
     (subactivity machine-block decomposition-27)
     (idef-process decomposition-27))


(and (doc produce-moulded-metal-item-1
      "The occurence of Produce-Moulded-Metal-Item in the Dec-27 schematic")
     (forall (?a : (activation-of ?a produce-moulded-metal-item-1))
      (activation-of ?a produce-moluded-metal-item))
     (forall (?a : (activation-of ?a produce-moulded-metal-item-1))
      (exists (?p : (activation-of ?p decomposition-27))
       (subactivity-occurrence ?a ?p)))
     (forall (?a : (activation-of ?a produce-moulded-metal-item-1))
      (activation-of ?a decomposition-1)))


(and (doc machine-block-1
      "The occurence of Machine-Block in the Dec-27 schematic")
     (forall (?a : (activation-of ?a machine-block-1))
      (activation-of ?a machine-block))
     (forall (?a : (activation-of ?a machine-block-1))
      (exists (?p : (activation-of ?p decomposition-27))
       (subactivity-occurrence ?a ?p))))


(and (doc l93 "L93")
     (follows produce-moluded-metal-item-1 machine-block-1
      decomposition-27))


(and (doc decomposition-1
      "Decomposition of Produce Moulded Metal Item")
     (subactivity change-mould-1 decomposition-1)
     (subactivity setup-furnace-1 decomposition-1)
     (subactivity analyse-metal-1 decomposition-1)
```

```
     (subactivity melt-1 decomposition-1)
     (subactivity mould-1 decomposition-1)
     (subactivity wait-1 decomposition-1)
     (subactivity clear-cavities-1 decomposition-1)
     (subactivity j8 decomposition-1) (subactivity j7 decomposition-1)
     (subactivity j6 decomposition-1) (subactivity j5 decomposition-1)
     (idef-process decomposition-1))


(and (doc change-mould-1
      "The occurence of Change-Mould in the Dec-1 schematic")
     (forall (?a : (activation-of ?a change-mould-1))
      (activation-of ?a change-mould))
     (forall (?a : (activation-of ?a change-mould-1))
      (exists (?p : (activation-of ?p decomposition-1))
       (subactivity-occurrence ?a ?p))))


(and (doc setup-furnace-1
      "The occurence of Setup-Furnace in the Dec-1 schematic")
     (forall (?a : (activation-of ?a setup-furnace-1))
      (activation-of ?a setup-furnace))
     (forall (?a : (activation-of ?a setup-furnace-1))
      (exists (?p : (activation-of ?p decomposition-1))
       (subactivity-occurrence ?a ?p))))


(and (doc analyse-metal-1
      "The occurence of Analyse-Metal in the Dec-1 schematic")
     (forall (?a : (activation-of ?a analyse-metal-1))
      (activation-of ?a analyse-metal))
     (forall (?a : (activation-of ?a analyse-metal-1))
      (exists (?p : (activation-of ?p decomposition-1))
       (subactivity-occurrence ?a ?p))))


(and (doc melt-1 "The occurence of Melt in the Dec-1 schematic")
     (forall (?a : (activation-of ?a melt-1)) (activation-of ?a melt))
     (forall (?a : (activation-of ?a melt-1))
      (exists (?p : (activation-of ?p decomposition-1))
       (subactivity-occurrence ?a ?p))))


(and (doc mould-1 "The occurence of Mould in the Dec-1 schematic")
     (forall (?a : (activation-of ?a mould-1))
      (activation-of ?a mould))
     (forall (?a : (activation-of ?a mould-1))
      (exists (?p : (activation-of ?p decomposition-1))
       (subactivity-occurrence ?a ?p)))
     (forall (?a : (activation-of ?a mould-1))
      (activation-of ?a decomposition-1-1)))


(and (doc wait-1 "The occurence of Wait in the Dec-1 schematic")
     (forall (?a : (activation-of ?a wait-1)) (activation-of ?a wait))
     (forall (?a : (activation-of ?a wait-1))
      (exists (?p : (activation-of ?p decomposition-1))
```

```
       (subactivity-occurrence ?a ?p))))


(and (doc clear-cavities-1
      "The occurence of Clear-Cavities in the Dec-1 schematic")
     (forall (?a : (activation-of ?a clear-cavities-1))
      (activation-of ?a clear-cavities))
     (forall (?a : (activation-of ?a clear-cavities-1))
      (exists (?p : (activation-of ?p decomposition-1))
       (subactivity-occurrence ?a ?p))))


(and (doc j8 "J8")
     (forall (?j : (activation-of ?j j8))
      (exists (?p : (activation-of ?p decomposition-1))
       (subactivity-occurrence ?j ?p)))
     (follows j8 mould-1 decomposition-1)
     (and (and_split j8 decomposition-1)
          (subactivity analyse-metal-1 j8) (subactivity melt-1 j8)))


(and (doc j7 "J7")
     (forall (?j : (activation-of ?j j7))
      (exists (?p : (activation-of ?p decomposition-1))
       (subactivity-occurrence ?j ?p)))
     (follows setup-furnace-1 j7 decomposition-1)
     (and (and_split j7 decomposition-1)
          (subactivity analyse-metal-1 j7) (subactivity melt-1 j7)))


(and (doc j6 "J6")
     (forall (?j : (activation-of ?j j6))
      (exists (?p : (activation-of ?p decomposition-1))
       (subactivity-occurrence ?j ?p)))
     (follows j6 setup-furnace-1 decomposition-1)
     (and (xor_split j6 decomposition-1) (subactivity j5 j6)
          (subactivity change-mould-1 j6)))


(and (doc j5 "J5")
     (forall (?j : (activation-of ?j j5))
      (exists (?p : (activation-of ?p decomposition-1))
       (subactivity-occurrence ?j ?p)))
     (and (and_split j5 decomposition-1) (subactivity j6 j5)
          (subactivity change-mould-1 j5)))


(and (doc l120 "L120") (follows mould-1 wait-1 decomposition-1))


(and (doc l123 "L123")
     (follows wait-1 clear-cavities-1 decomposition-1))


(and (doc decomposition-1-1 "Decomposition of Mould")
     (subactivity setup-racks-1 decomposition-1-1)
     (subactivity pour-1 decomposition-1-1)
```

```
     (subactivity remove-racks-1 decomposition-1-1)
     (subactivity batch-complete-1 decomposition-1-1)
     (idef-process decomposition-1-1))


(and (doc setup-racks-1
      "The occurence of Setup-Racks in the Dec-1-1 schematic")
     (forall (?a : (activation-of ?a setup-racks-1))
      (activation-of ?a setup-racks))
     (forall (?a : (activation-of ?a setup-racks-1))
      (exists (?p : (activation-of ?p decomposition-1-1))
       (subactivity-occurrence ?a ?p))))


(and (doc pour-1 "The occurence of Pour in the Dec-1-1 schematic")
     (forall (?a : (activation-of ?a pour-1)) (activation-of ?a pour))
     (forall (?a : (activation-of ?a pour-1))
      (exists (?p : (activation-of ?p decomposition-1-1))
       (subactivity-occurrence ?a ?p))))


(and (doc remove-racks-1
      "The occurence of Remove-Racks in the Dec-1-1 schematic")
     (forall (?a : (activation-of ?a remove-racks-1))
      (activation-of ?a remove-racks))
     (forall (?a : (activation-of ?a remove-racks-1))
      (exists (?p : (activation-of ?p decomposition-1-1))
       (subactivity-occurrence ?a ?p))))


(and (doc batch-complete-1
      "The occurence of Batch-Complete in the Dec-1-1 schematic")
     (forall (?a : (activation-of ?a batch-complete-1))
      (activation-of ?a batch-complete))
     (forall (?a : (activation-of ?a batch-complete-1))
      (exists (?p : (activation-of ?p decomposition-1-1))
       (subactivity-occurrence ?a ?p))))
```

A.3 PSL simplified KIF representation

```
(doc make-gt350 "Make GT350")


(and (doc make-interior "Make Interior")
     (forall ?a
      (=> (activation-of ?a make-interior)
       (exists ?o1 (=> (instance-of ?o1 a002-bench) (in ?o1 ?a))))))


(and (doc make-drive "Make Drive")
     (forall ?a
      (=> (activation-of ?a make-drive)
       (exists ?o1 (=> (instance-of ?o1 a003-bench) (in ?o1 ?a))))))


(and (doc make-trim "Make Trim")
     (forall ?a
      (=> (activation-of ?a make-trim)
       (exists ?o1
        (exists ?o2
         (=>
          (and (instance-of ?o1 a002-bench)
               (instance-of ?o2 a005-bench))
          (and (in ?o1 ?a) (in ?o2 ?a))))))))


(doc make-chassis "Make Chassis")


(doc final-assembly "Final Assembly")


(doc make-block "Make Block")


(doc make-hamess "Make Hamess")


(doc make-wires "Make Wires")


(doc assemble-engine "Assemble Engine")


(doc produce-moulded-metal-item "Produce Moulded Metal Item")


(doc machine-block "Machine Block")


(doc change-mould "Change Mould")


(doc setup-furnace "Setup Furnace")
```

```
(doc analyse-metal "Analyse Metal")


(doc melt "Melt")


(doc mould "Mould")


(doc wait "Wait")


(doc clear-cavities "Clear Cavities")


(doc setup-racks "Setup Racks")


(doc pour "Pour")


(doc remove-racks "Remove Racks")


(doc batch-complete "Batch Complete")


(and (doc make-gt350-proc "The Top Level GT350 process schematic")
     (and (subactivity make-gt350-1 make-gt350-proc)
          (idef-process make-gt350-proc)))


(and (doc make-gt-350-1 "The occurence of Make-GT350 in the schematic")
     (and (forall ?a
           (=> (activation-of ?a make-gt-350-1)
            (activation-of ?a make-gt350)))
          (and (forall ?a
                (=> (activation-of ?a make-gt-350-1)
                 (activation-of ?a decomposition-19)))
               (duration make_engine-1 5))))


(and (doc decomposition-19 "Decomposition of Make GT350")
     (and (subactivity make-interior-1 decomposition-19)
          (and (subactivity make-drive-1 decomposition-19)
               (and (subactivity make-trim-1 decomposition-19)
                    (and (subactivity make-engine-1 decomposition-19)
                         (and (subactivity make-chassis-1
                                decomposition-19)
                              (and (subactivity final-assembly-1
                                     decomposition-19)
                                   (and (subactivity j2
                                          decomposition-19)
                                        (and (subactivity j1
                                               decomposition-19)
                                             (idef-process
```

```
                                              decomposition-19))))))))))


(and (doc make-interior-1
      "The occurence of Make-Interior in the Dec-19 schematic")
     (and (forall ?a
           (=> (activation-of ?a make-interior-1)
            (activation-of ?a make-interior)))
          (forall ?a
           (=> (activation-of ?a make-interior-1)
            (exists ?p
             (=> (activation-of ?p decomposition-19)
              (subactivity-occurrence ?a ?p)))))))


(and (doc make-drive-1
      "The occurence of Make-Drive in the Dec-19 schematic")
     (and (forall ?a
           (=> (activation-of ?a make-drive-1)
            (activation-of ?a make-drive)))
          (forall ?a
           (=> (activation-of ?a make-drive-1)
            (exists ?p
             (=> (activation-of ?p decomposition-19)
              (subactivity-occurrence ?a ?p)))))))


(and (doc make-trim-1
      "The occurence of Make-Trim in the Dec-19 schematic")
     (and (forall ?a
           (=> (activation-of ?a make-trim-1)
            (activation-of ?a make-trim)))
          (forall ?a
           (=> (activation-of ?a make-trim-1)
            (exists ?p
             (=> (activation-of ?p decomposition-19)
              (subactivity-occurrence ?a ?p)))))))


(and (doc make-engine-1
      "The occurence of Make-Engine in the Dec-19 schematic")
     (and (forall ?a
           (=> (activation-of ?a make-engine-1)
            (activation-of ?a make-engine)))
          (and (forall ?a
                (=> (activation-of ?a make-engine-1)
                 (exists ?p
                  (=> (activation-of ?p decomposition-19)
                   (subactivity-occurrence ?a ?p)))))
               (forall ?a
                (=> (activation-of ?a make-engine-1)
                 (activation-of ?a decomposition-26))))))


(and (doc make-chassis-1
      "The occurence of Make-Chassis in the Dec-19 schematic")
     (and (forall ?a
```

```
          (=> (activation-of ?a make-chassis-1)
           (activation-of ?a make-chassis)))
         (forall ?a
          (=> (activation-of ?a make-chassis-1)
           (exists ?p
            (=> (activation-of ?p decomposition-19)
             (subactivity-occurrence ?a ?p)))))))


(and (doc final-assembly-1
      "The occurence of Final-Assembly in the Dec-19 schematic")
     (and (forall ?a
          (=> (activation-of ?a final-assembly-1)
           (activation-of ?a final-assembly)))
         (forall ?a
          (=> (activation-of ?a final-assembly-1)
           (exists ?p
            (=> (activation-of ?p decomposition-19)
             (subactivity-occurrence ?a ?p)))))))


(and (doc j2 "J2")
     (and (forall ?j
          (=> (activation-of ?j j2)
           (exists ?p
            (=> (activation-of ?p decomposition-19)
             (subactivity-occurrence ?j ?p)))))
         (and (follows j2 final-assembly-1 decomposition-19)
              (and (and_split j2 decomposition-19)
                   (and (subactivity make-interior-1 j2)
                        (and (subactivity make-drive-1 j2)
                             (and (subactivity make-trim-1 j2)
                                  (and (subactivity make-engine-1 j2)
                                       (subactivity make-chassis-1
                                        j2)))))))))


(and (doc j1 "J1")
     (and (forall ?j
          (=> (activation-of ?j j1)
           (exists ?p
            (=> (activation-of ?p decomposition-19)
             (subactivity-occurrence ?j ?p)))))
         (and (and_split j1 decomposition-19)
              (and (subactivity make-interior-1 j1)
                   (and (subactivity make-drive-1 j1)
                        (and (subactivity make-trim-1 j1)
                             (and (subactivity make-engine-1 j1)
                                  (subactivity make-chassis-1
                                   j1))))))))


(and (doc decomposition-26 "Decomposition of Make Engine")
     (and (subactivity make-block-1 decomposition-26)
          (and (subactivity make-harness-1 decomposition-26)
               (and (subactivity make-wires-1 decomposition-26)
                    (and (subactivity assemble-engine-1
```

```
                            decomposition-26)
                           (and (subactivity j4 decomposition-26)
                                (and (subactivity j3 decomposition-26)
                                     (idef-process
                                      decomposition-26))))))))


(and (doc make-block-1
      "The occurence of Make-Block in the Dec-26 schematic")
     (and (forall ?a
           (=> (activation-of ?a make-block-1)
            (activation-of ?a make-block)))
          (and (forall ?a
                (=> (activation-of ?a make-block-1)
                 (exists ?p
                  (=> (activation-of ?p decomposition-26)
                   (subactivity-occurrence ?a ?p)))))
               (forall ?a
                (=> (activation-of ?a make-block-1)
                 (activation-of ?a decomposition-27))))))


(and (doc make-harness-1
      "The occurence of Make-Harness in the Dec-26 schematic")
     (and (forall ?a
           (=> (activation-of ?a make-harness-1)
            (activation-of ?a make-harness)))
          (forall ?a
           (=> (activation-of ?a make-harness-1)
            (exists ?p
             (=> (activation-of ?p decomposition-26)
              (subactivity-occurrence ?a ?p)))))))


(and (doc make-wires-1
      "The occurence of Make-Wires in the Dec-26 schematic")
     (and (forall ?a
           (=> (activation-of ?a make-wires-1)
            (activation-of ?a make-wires)))
          (forall ?a
           (=> (activation-of ?a make-wires-1)
            (exists ?p
             (=> (activation-of ?p decomposition-26)
              (subactivity-occurrence ?a ?p)))))))


(and (doc assemble-engine-1
      "The occurence of Assemble-Engine in the Dec-26 schematic")
     (and (forall ?a
           (=> (activation-of ?a assemble-engine-1)
            (activation-of ?a assemble-engine)))
          (forall ?a
           (=> (activation-of ?a assemble-engine-1)
            (exists ?p
             (=> (activation-of ?p decomposition-26)
              (subactivity-occurrence ?a ?p)))))))
```

```
(and (doc j4 "J4")
     (and (forall ?j
           (=> (activation-of ?j j4)
            (exists ?p
             (=> (activation-of ?p decomposition-26)
              (subactivity-occurrence ?j ?p)))))
          (and (follows j4 assemble-engine-1 decomposition-26)
               (and (and_split j4 decomposition-26)
                    (and (subactivity make-block-1 j4)
                         (and (subactivity make-harness-1 j4)
                              (subactivity make-wires-1 j4)))))))


(and (doc j3 "J3")
     (and (forall ?j
           (=> (activation-of ?j j3)
            (exists ?p
             (=> (activation-of ?p decomposition-26)
              (subactivity-occurrence ?j ?p)))))
          (and (and_split j3 decomposition-26)
               (and (subactivity make-block-1 j3)
                    (and (subactivity make-harness-1 j3)
                         (subactivity make-wires-1 j3))))))


(and (doc decomposition-27 "Decomposition of Make Block")
     (and (subactivity produce-moulded-metal-item decomposition-27)
          (and (subactivity machine-block decomposition-27)
               (idef-process decomposition-27))))


(and (doc produce-moulded-metal-item-1
      "The occurence of Produce-Moulded-Metal-Item in the Dec-27 schematic")
     (and (forall ?a
           (=> (activation-of ?a produce-moulded-metal-item-1)
            (activation-of ?a produce-moluded-metal-item)))
          (and (forall ?a
                (=> (activation-of ?a produce-moulded-metal-item-1)
                 (exists ?p
                  (=> (activation-of ?p decomposition-27)
                   (subactivity-occurrence ?a ?p)))))
               (forall ?a
                (=> (activation-of ?a produce-moulded-metal-item-1)
                 (activation-of ?a decomposition-1))))))


(and (doc machine-block-1
      "The occurence of Machine-Block in the Dec-27 schematic")
     (and (forall ?a
           (=> (activation-of ?a machine-block-1)
            (activation-of ?a machine-block)))
          (forall ?a
           (=> (activation-of ?a machine-block-1)
            (exists ?p
             (=> (activation-of ?p decomposition-27)
              (subactivity-occurrence ?a ?p)))))))
```

```
(and (doc l93 "L93")
     (follows produce-moluded-metal-item-1 machine-block-1
      decomposition-27))


(and (doc decomposition-1
      "Decomposition of Produce Moulded Metal Item")
     (and (subactivity change-mould-1 decomposition-1)
          (and (subactivity setup-furnace-1 decomposition-1)
               (and (subactivity analyse-metal-1 decomposition-1)
                    (and (subactivity melt-1 decomposition-1)
                         (and (subactivity mould-1 decomposition-1)
                              (and (subactivity wait-1 decomposition-1)
                                   (and (subactivity clear-cavities-1
                                         decomposition-1)
                                        (and (subactivity j8
                                              decomposition-1)
                                             (and (subactivity j7
                                                   decomposition-1)
                                                  (and (subactivity j6
                                                        decomposition-1)
                                                       (and (subactivity
                                                             j5
                                                             decomposition-1)
                                                            (idef-process
                                                             decomposition-1)))))))))))))


(and (doc change-mould-1
      "The occurence of Change-Mould in the Dec-1 schematic")
     (and (forall ?a
           (=> (activation-of ?a change-mould-1)
            (activation-of ?a change-mould)))
          (forall ?a
           (=> (activation-of ?a change-mould-1)
            (exists ?p
             (=> (activation-of ?p decomposition-1)
              (subactivity-occurrence ?a ?p)))))))


(and (doc setup-furnace-1
      "The occurence of Setup-Furnace in the Dec-1 schematic")
     (and (forall ?a
           (=> (activation-of ?a setup-furnace-1)
            (activation-of ?a setup-furnace)))
          (forall ?a
           (=> (activation-of ?a setup-furnace-1)
            (exists ?p
             (=> (activation-of ?p decomposition-1)
              (subactivity-occurrence ?a ?p)))))))


(and (doc analyse-metal-1
      "The occurence of Analyse-Metal in the Dec-1 schematic")
     (and (forall ?a
```

```
           (=> (activation-of ?a analyse-metal-1)
            (activation-of ?a analyse-metal)))
          (forall ?a
           (=> (activation-of ?a analyse-metal-1)
            (exists ?p
             (=> (activation-of ?p decomposition-1)
              (subactivity-occurrence ?a ?p)))))))


(and (doc melt-1 "The occurence of Melt in the Dec-1 schematic")
     (and (forall ?a
           (=> (activation-of ?a melt-1) (activation-of ?a melt)))
          (forall ?a
           (=> (activation-of ?a melt-1)
            (exists ?p
             (=> (activation-of ?p decomposition-1)
              (subactivity-occurrence ?a ?p)))))))


(and (doc mould-1 "The occurence of Mould in the Dec-1 schematic")
     (and (forall ?a
           (=> (activation-of ?a mould-1) (activation-of ?a mould)))
          (and (forall ?a
                (=> (activation-of ?a mould-1)
                 (exists ?p
                  (=> (activation-of ?p decomposition-1)
                   (subactivity-occurrence ?a ?p)))))
               (forall ?a
                (=> (activation-of ?a mould-1)
                 (activation-of ?a decomposition-1-1))))))


(and (doc wait-1 "The occurence of Wait in the Dec-1 schematic")
     (and (forall ?a
           (=> (activation-of ?a wait-1) (activation-of ?a wait)))
          (forall ?a
           (=> (activation-of ?a wait-1)
            (exists ?p
             (=> (activation-of ?p decomposition-1)
              (subactivity-occurrence ?a ?p)))))))


(and (doc clear-cavities-1
      "The occurence of Clear-Cavities in the Dec-1 schematic")
     (and (forall ?a
           (=> (activation-of ?a clear-cavities-1)
            (activation-of ?a clear-cavities)))
          (forall ?a
           (=> (activation-of ?a clear-cavities-1)
            (exists ?p
             (=> (activation-of ?p decomposition-1)
              (subactivity-occurrence ?a ?p)))))))


(and (doc j8 "J8")
     (and (forall ?j
           (=> (activation-of ?j j8)
```

```
          (exists ?p
           (=> (activation-of ?p decomposition-1)
            (subactivity-occurrence ?j ?p)))))
         (and (follows j8 mould-1 decomposition-1)
              (and (and_split j8 decomposition-1)
                   (and (subactivity analyse-metal-1 j8)
                        (subactivity melt-1 j8))))))


(and (doc j7 "J7")
     (and (forall ?j
          (=> (activation-of ?j j7)
           (exists ?p
            (=> (activation-of ?p decomposition-1)
             (subactivity-occurrence ?j ?p)))))
          (and (follows setup-furnace-1 j7 decomposition-1)
               (and (and_split j7 decomposition-1)
                    (and (subactivity analyse-metal-1 j7)
                         (subactivity melt-1 j7))))))


(and (doc j6 "J6")
     (and (forall ?j
          (=> (activation-of ?j j6)
           (exists ?p
            (=> (activation-of ?p decomposition-1)
             (subactivity-occurrence ?j ?p)))))
          (and (follows j6 setup-furnace-1 decomposition-1)
               (and (xor_split j6 decomposition-1)
                    (and (subactivity j5 j6)
                         (subactivity change-mould-1 j6))))))


(and (doc j5 "J5")
     (and (forall ?j
          (=> (activation-of ?j j5)
           (exists ?p
            (=> (activation-of ?p decomposition-1)
             (subactivity-occurrence ?j ?p)))))
          (and (and_split j5 decomposition-1)
               (and (subactivity j6 j5)
                    (subactivity change-mould-1 j5)))))


(and (doc l120 "L120") (follows mould-1 wait-1 decomposition-1))


(and (doc l123 "L123")
     (follows wait-1 clear-cavities-1 decomposition-1))


(and (doc decomposition-1-1 "Decomposition of Mould")
     (and (subactivity setup-racks-1 decomposition-1-1)
          (and (subactivity pour-1 decomposition-1-1)
               (and (subactivity remove-racks-1 decomposition-1-1)
                    (and (subactivity batch-complete-1
                          decomposition-1-1)
```

```
                         (idef-process decomposition-1-1))))))


(and (doc setup-racks-1
      "The occurence of Setup-Racks in the Dec-1-1 schematic")
     (and (forall ?a
           (=> (activation-of ?a setup-racks-1)
            (activation-of ?a setup-racks)))
          (forall ?a
           (=> (activation-of ?a setup-racks-1)
            (exists ?p
             (=> (activation-of ?p decomposition-1-1)
              (subactivity-occurrence ?a ?p)))))))


(and (doc pour-1 "The occurence of Pour in the Dec-1-1 schematic")
     (and (forall ?a
           (=> (activation-of ?a pour-1) (activation-of ?a pour)))
          (forall ?a
           (=> (activation-of ?a pour-1)
            (exists ?p
             (=> (activation-of ?p decomposition-1-1)
              (subactivity-occurrence ?a ?p)))))))


(and (doc remove-racks-1
      "The occurence of Remove-Racks in the Dec-1-1 schematic")
     (and (forall ?a
           (=> (activation-of ?a remove-racks-1)
            (activation-of ?a remove-racks)))
          (forall ?a
           (=> (activation-of ?a remove-racks-1)
            (exists ?p
             (=> (activation-of ?p decomposition-1-1)
              (subactivity-occurrence ?a ?p)))))))


(and (doc batch-complete-1
      "The occurence of Batch-Complete in the Dec-1-1 schematic")
     (and (forall ?a
           (=> (activation-of ?a batch-complete-1)
            (activation-of ?a batch-complete)))
          (forall ?a
           (=> (activation-of ?a batch-complete-1)
            (exists ?p
             (=> (activation-of ?p decomposition-1-1)
              (subactivity-occurrence ?a ?p)))))))
```

A.4 PSL Prolog representation

```
doc(make_gt350, "Make GT350").

in(Z01, ZV476) :- instance_of(Z01, a002_bench), activation_of(ZV476, make_interior).

doc(make_interior, "Make Interior").

in(Z01, ZV479) :- instance_of(Z01, a003_bench), activation_of(ZV479, make_drive).

doc(make_drive, "Make Drive").

in(Z02, ZV482) :- activation_of(ZV482, make_trim), instance_of(Z02, a005_bench),
instance_of(Z01, a002_bench).

in(Z01, ZV482) :- activation_of(ZV482, make_trim), instance_of(Z02, a005_bench),
instance_of(Z01, a002_bench).

doc(make_trim, "Make Trim").

doc(make_chassis, "Make Chassis").

doc(final_assembly, "Final Assembly").

doc(make_block, "Make Block").

doc(make_hamess, "Make Hamess").

doc(make_wires, "Make Wires").

doc(assemble_engine, "Assemble Engine").

doc(produce_moulded_metal_item, "Produce Moulded Metal Item").

doc(machine_block, "Machine Block").

doc(change_mould, "Change Mould").

doc(setup_furnace, "Setup Furnace").

doc(analyse_metal, "Analyse Metal").

doc(melt, "Melt").

doc(mould, "Mould").

doc(wait, "Wait").

doc(clear_cavities, "Clear Cavities").

doc(setup_racks, "Setup Racks").

doc(pour, "Pour").

doc(remove_racks, "Remove Racks").

doc(batch_complete, "Batch Complete").

idef_process(make_gt350_proc).
```

```
subactivity(make_gt350_1, make_gt350_proc).

doc(make_gt350_proc, "The Top Level GT350 process schematic").

duration(make_engine_1, 5).

activation_of(ZV488, decomposition_19) :- activation_of(ZV488, make_gt_350_1).

activation_of(ZV487, make_gt350) :- activation_of(ZV487, make_gt_350_1).

doc(make_gt_350_1, "The occurence of Make-GT350 in the schematic").

idef_process(decomposition_19).

subactivity(j1, decomposition_19).

subactivity(j2, decomposition_19).

subactivity(final_assembly_1, decomposition_19).

subactivity(make_chassis_1, decomposition_19).

subactivity(make_engine_1, decomposition_19).

subactivity(make_trim_1, decomposition_19).

subactivity(make_drive_1, decomposition_19).

subactivity(make_interior_1, decomposition_19).

doc(decomposition_19, "Decomposition of Make GT350").

subactivity_occurrence(ZV490, sk492(ZV489, ZV490)) :- activation_of(sk492(ZV489, ZV490),
decomposition_19), activation_of(ZV490, make_interior_1).

activation_of(ZV489, make_interior) :- activation_of(ZV489, make_interior_1).

doc(make_interior_1, "The occurence of Make-Interior in the Dec-19 schematic").

subactivity_occurrence(ZV494, sk496(ZV493, ZV494)) :- activation_of(sk496(ZV493, ZV494),
decomposition_19), activation_of(ZV494, make_drive_1).

activation_of(ZV493, make_drive) :- activation_of(ZV493, make_drive_1).

doc(make_drive_1, "The occurence of Make-Drive in the Dec-19 schematic").

subactivity_occurrence(ZV498, sk500(ZV497, ZV498)) :- activation_of(sk500(ZV497, ZV498),
decomposition_19), activation_of(ZV498, make_trim_1).

activation_of(ZV497, make_trim) :- activation_of(ZV497, make_trim_1).

doc(make_trim_1, "The occurence of Make-Trim in the Dec-19 schematic").

activation_of(ZV504, decomposition_26) :- activation_of(ZV504, make_engine_1).

subactivity_occurrence(ZV502, sk505(ZV501, ZV502)) :- activation_of(sk505(ZV501, ZV502),
```

```
decomposition_19), activation_of(ZV502, make_engine_1).

activation_of(ZV501, make_engine) :- activation_of(ZV501, make_engine_1).

doc(make_engine_1, "The occurence of Make-Engine in the Dec-19 schematic").

subactivity_occurrence(ZV507, sk509(ZV506, ZV507)) :- activation_of(sk509(ZV506, ZV507),
decomposition_19), activation_of(ZV507, make_chassis_1).

activation_of(ZV506, make_chassis) :- activation_of(ZV506, make_chassis_1).

doc(make_chassis_1, "The occurence of Make-Chassis in the Dec-19 schematic").

subactivity_occurrence(ZV511, sk513(ZV510, ZV511)) :- activation_of(sk513(ZV510, ZV511),
decomposition_19), activation_of(ZV511, final_assembly_1).

activation_of(ZV510, final_assembly) :- activation_of(ZV510, final_assembly_1).

doc(final_assembly_1, "The occurence of Final-Assembly in the Dec-19 schematic").

subactivity(make_chassis_1, j2).

subactivity(make_engine_1, j2).

subactivity(make_trim_1, j2).

subactivity(make_drive_1, j2).

subactivity(make_interior_1, j2).

and_split(j2, decomposition_19).

follows(j2, final_assembly_1, decomposition_19).

subactivity_occurrence(ZV514, sk516(ZV514)) :- activation_of(sk516(ZV514),
decomposition_19), activation_of(ZV514, j2).

doc(j2, "J2").

subactivity(make_chassis_1, j1).

subactivity(make_engine_1, j1).

subactivity(make_trim_1, j1).

subactivity(make_drive_1, j1).

subactivity(make_interior_1, j1).

and_split(j1, decomposition_19).

subactivity_occurrence(ZV517, sk519(ZV517)) :- activation_of(sk519(ZV517),
decomposition_19), activation_of(ZV517, j1).

doc(j1, "J1").

idef_process(decomposition_26).
```

```
subactivity(j3, decomposition_26).

subactivity(j4, decomposition_26).

subactivity(assemble_engine_1, decomposition_26).

subactivity(make_wires_1, decomposition_26).

subactivity(make_harness_1, decomposition_26).

subactivity(make_block_1, decomposition_26).

doc(decomposition_26, "Decomposition of Make Engine").

activation_of(ZV523, decomposition_27) :- activation_of(ZV523, make_block_1).

subactivity_occurrence(ZV521, sk524(ZV520, ZV521)) :- activation_of(sk524(ZV520, ZV521),
decomposition_26), activation_of(ZV521, make_block_1).

activation_of(ZV520, make_block) :- activation_of(ZV520, make_block_1).

doc(make_block_1, "The occurence of Make-Block in the Dec-26 schematic").

subactivity_occurrence(ZV526, sk528(ZV525, ZV526)) :- activation_of(sk528(ZV525, ZV526),
decomposition_26), activation_of(ZV526, make_harness_1).

activation_of(ZV525, make_harness) :- activation_of(ZV525, make_harness_1).

doc(make_harness_1, "The occurence of Make-Harness in the Dec-26 schematic").

subactivity_occurrence(ZV530, sk532(ZV529, ZV530)) :- activation_of(sk532(ZV529, ZV530),
decomposition_26), activation_of(ZV530, make_wires_1).

activation_of(ZV529, make_wires) :- activation_of(ZV529, make_wires_1).

doc(make_wires_1, "The occurence of Make-Wires in the Dec-26 schematic").

subactivity_occurrence(ZV534, sk536(ZV533, ZV534)) :- activation_of(sk536(ZV533, ZV534),
decomposition_26), activation_of(ZV534, assemble_engine_1).

activation_of(ZV533, assemble_engine) :- activation_of(ZV533, assemble_engine_1).

doc(assemble_engine_1, "The occurence of Assemble-Engine in the Dec-26 schematic").

subactivity(make_wires_1, j4).

subactivity(make_harness_1, j4).

subactivity(make_block_1, j4).

and_split(j4, decomposition_26).

follows(j4, assemble_engine_1, decomposition_26).

subactivity_occurrence(ZV537, sk539(ZV537)) :- activation_of(sk539(ZV537),
decomposition_26), activation_of(ZV537, j4).
```

```
doc(j4, "J4").

subactivity(make_wires_1, j3).

subactivity(make_harness_1, j3).

subactivity(make_block_1, j3).

and_split(j3, decomposition_26).

subactivity_occurrence(ZV540, sk542(ZV540)) :- activation_of(sk542(ZV540),
decomposition_26), activation_of(ZV540, j3).

doc(j3, "J3").

idef_process(decomposition_27).

subactivity(machine_block, decomposition_27).

subactivity(produce_moulded_metal_item, decomposition_27).

doc(decomposition_27, "Decomposition of Make Block").

activation_of(ZV546, decomposition_1) :- activation_of(ZV546,
produce_moulded_metal_item_1).

subactivity_occurrence(ZV544, sk547(ZV543, ZV544)) :- activation_of(sk547(ZV543, ZV544),
decomposition_27), activation_of(ZV544, produce_moulded_metal_item_1).

activation_of(ZV543, produce_moluded_metal_item) :- activation_of(ZV543,
produce_moulded_metal_item_1).

doc(produce_moulded_metal_item_1,
"The occurence of Produce-Moulded-Metal-Item in the Dec-27 schematic").

subactivity_occurrence(ZV549, sk551(ZV548, ZV549)) :- activation_of(sk551(ZV548, ZV549),
decomposition_27), activation_of(ZV549, machine_block_1).

activation_of(ZV548, machine_block) :- activation_of(ZV548, machine_block_1).

doc(machine_block_1, "The occurence of Machine-Block in the Dec-27 schematic").

follows(produce_moluded_metal_item_1, machine_block_1, decomposition_27).

doc(l93, "L93").

idef_process(decomposition_1).

subactivity(j5, decomposition_1).

subactivity(j6, decomposition_1).

subactivity(j7, decomposition_1).

subactivity(j8, decomposition_1).
```

```
subactivity(clear_cavities_1, decomposition_1).

subactivity(wait_1, decomposition_1).

subactivity(mould_1, decomposition_1).

subactivity(melt_1, decomposition_1).

subactivity(analyse_metal_1, decomposition_1).

subactivity(setup_furnace_1, decomposition_1).

subactivity(change_mould_1, decomposition_1).

doc(decomposition_1, "Decomposition of Produce Moulded Metal Item").

subactivity_occurrence(ZV553, sk555(ZV552, ZV553)) :- activation_of(sk555(ZV552, ZV553),
decomposition_1), activation_of(ZV553, change_mould_1).

activation_of(ZV552, change_mould) :- activation_of(ZV552, change_mould_1).

doc(change_mould_1, "The occurence of Change-Mould in the Dec-1 schematic").

subactivity_occurrence(ZV557, sk559(ZV556, ZV557)) :- activation_of(sk559(ZV556, ZV557),
decomposition_1), activation_of(ZV557, setup_furnace_1).

activation_of(ZV556, setup_furnace) :- activation_of(ZV556, setup_furnace_1).

doc(setup_furnace_1, "The occurence of Setup-Furnace in the Dec-1 schematic").

subactivity_occurrence(ZV561, sk563(ZV560, ZV561)) :- activation_of(sk563(ZV560, ZV561),
decomposition_1), activation_of(ZV561, analyse_metal_1).

activation_of(ZV560, analyse_metal) :- activation_of(ZV560, analyse_metal_1).

doc(analyse_metal_1, "The occurence of Analyse-Metal in the Dec-1 schematic").

subactivity_occurrence(ZV565, sk567(ZV564, ZV565)) :- activation_of(sk567(ZV564, ZV565),
decomposition_1), activation_of(ZV565, melt_1).

activation_of(ZV564, melt) :- activation_of(ZV564, melt_1).

doc(melt_1, "The occurence of Melt in the Dec-1 schematic").

activation_of(ZV571, decomposition_1_1) :- activation_of(ZV571, mould_1).

subactivity_occurrence(ZV569, sk572(ZV568, ZV569)) :- activation_of(sk572(ZV568, ZV569),
decomposition_1), activation_of(ZV569, mould_1).

activation_of(ZV568, mould) :- activation_of(ZV568, mould_1).

doc(mould_1, "The occurence of Mould in the Dec-1 schematic").

subactivity_occurrence(ZV574, sk576(ZV573, ZV574)) :- activation_of(sk576(ZV573, ZV574),
decomposition_1), activation_of(ZV574, wait_1).

activation_of(ZV573, wait) :- activation_of(ZV573, wait_1).
```

```
doc(wait_1, "The occurence of Wait in the Dec-1 schematic").

subactivity_occurrence(ZV578, sk580(ZV577, ZV578)) :- activation_of(sk580(ZV577, ZV578),
decomposition_1), activation_of(ZV578, clear_cavities_1).

activation_of(ZV577, clear_cavities) :- activation_of(ZV577, clear_cavities_1).

doc(clear_cavities_1, "The occurence of Clear-Cavities in the Dec-1 schematic").

subactivity(melt_1, j8).

subactivity(analyse_metal_1, j8).

and_split(j8, decomposition_1).

follows(j8, mould_1, decomposition_1).

subactivity_occurrence(ZV581, sk583(ZV581)) :- activation_of(sk583(ZV581),
decomposition_1), activation_of(ZV581, j8).

doc(j8, "J8").

subactivity(melt_1, j7).

subactivity(analyse_metal_1, j7).

and_split(j7, decomposition_1).

follows(setup_furnace_1, j7, decomposition_1).

subactivity_occurrence(ZV584, sk586(ZV584)) :- activation_of(sk586(ZV584),
decomposition_1), activation_of(ZV584, j7).

doc(j7, "J7").

subactivity(change_mould_1, j6).

subactivity(j5, j6).

xor_split(j6, decomposition_1).

follows(j6, setup_furnace_1, decomposition_1).

subactivity_occurrence(ZV587, sk589(ZV587)) :- activation_of(sk589(ZV587),
decomposition_1), activation_of(ZV587, j6).

doc(j6, "J6").

subactivity(change_mould_1, j5).

subactivity(j6, j5).

and_split(j5, decomposition_1).

subactivity_occurrence(ZV590, sk592(ZV590)) :- activation_of(sk592(ZV590),
decomposition_1), activation_of(ZV590, j5).
```

```
doc(j5, "J5").

follows(mould_1, wait_1, decomposition_1).

doc(l120, "L120").

follows(wait_1, clear_cavities_1, decomposition_1).

doc(l123, "L123").

idef_process(decomposition_1_1).

subactivity(batch_complete_1, decomposition_1_1).

subactivity(remove_racks_1, decomposition_1_1).

subactivity(pour_1, decomposition_1_1).

subactivity(setup_racks_1, decomposition_1_1).

doc(decomposition_1_1, "Decomposition of Mould").

subactivity_occurrence(ZV594, sk596(ZV593, ZV594)) :- activation_of(sk596(ZV593, ZV594),
decomposition_1_1), activation_of(ZV594, setup_racks_1).

activation_of(ZV593, setup_racks) :- activation_of(ZV593, setup_racks_1).

doc(setup_racks_1, "The occurence of Setup-Racks in the Dec-1-1 schematic").

subactivity_occurrence(ZV598, sk600(ZV597, ZV598)) :- activation_of(sk600(ZV597, ZV598),
decomposition_1_1), activation_of(ZV598, pour_1).

activation_of(ZV597, pour) :- activation_of(ZV597, pour_1).

doc(pour_1, "The occurence of Pour in the Dec-1-1 schematic").

subactivity_occurrence(ZV602, sk604(ZV601, ZV602)) :- activation_of(sk604(ZV601, ZV602),
decomposition_1_1), activation_of(ZV602, remove_racks_1).

activation_of(ZV601, remove_racks) :- activation_of(ZV601, remove_racks_1).

doc(remove_racks_1, "The occurence of Remove-Racks in the Dec-1-1 schematic").

subactivity_occurrence(ZV606, sk608(ZV605, ZV606)) :- activation_of(sk608(ZV605, ZV606),
decomposition_1_1), activation_of(ZV606, batch_complete_1).

activation_of(ZV605, batch_complete) :- activation_of(ZV605, batch_complete_1).

doc(batch_complete_1, "The occurence of Batch-Complete in the Dec-1-1 schematic").
```

References

[1] Michael R. Genesereth and Richard E. Fikes. Knowledge Interchange Format Version 3.0 Reference Manual. Technical report, Logic Group, Stanford University, CA., 1992.

[2] Knowledge Based Systems Inc. Foundations for Product Realization Process Knowledge Sharing. Technical report, U.S. Department of Commerce, NOAA Contract No. 50-DKNB-7-90095.

[3] Richard J. Mayer, Christopher P. Menzel, Michael K. Painter, Paula S. deWitte, Thomas Blinn, and Benjamin Perakath. Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report. Technical report, Knowledge Based Systems Inc., KBSI-IICE-90-STR-01-0592-02, 1995.

[4] Stephen T. Polyak and Stuart Aitken. Manufacturing Process Interoperability Scenario. Technical report, National Institute of Standards and Technology, 1998.

[5] Stephen T. Polyak, Michael Gruninger, Jintae Lee, and Chris Menzel. Applying the Process Interchange Format (PIF) to a Supply Chain Process Interoperability Scenario, 1998.