Compositional verification of interlocking Status and discussion

Christophe Limbrée

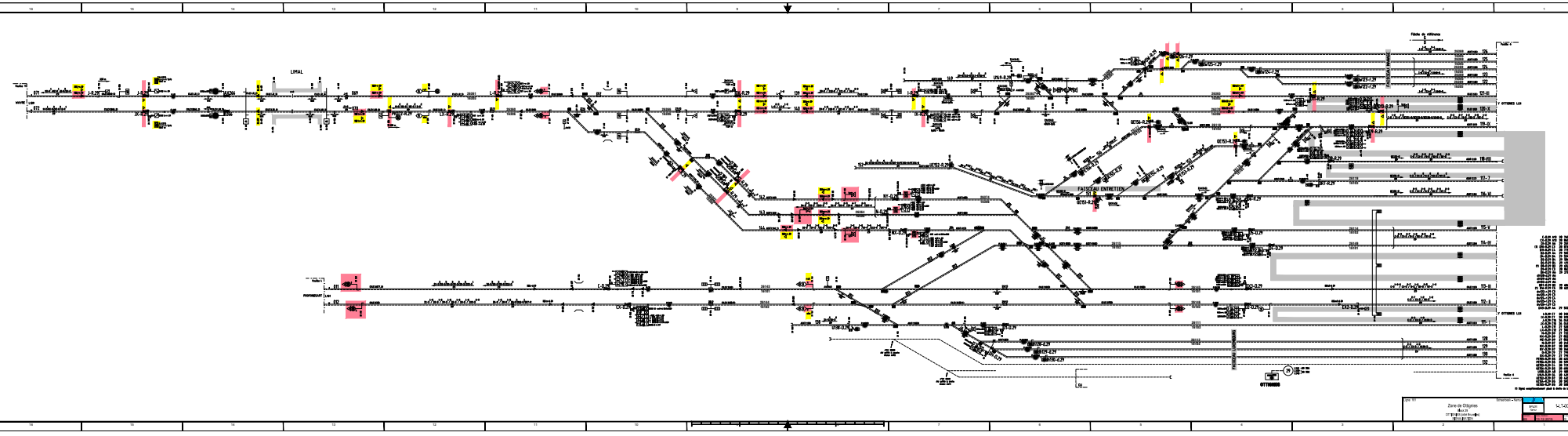
INGI

2017-04-21

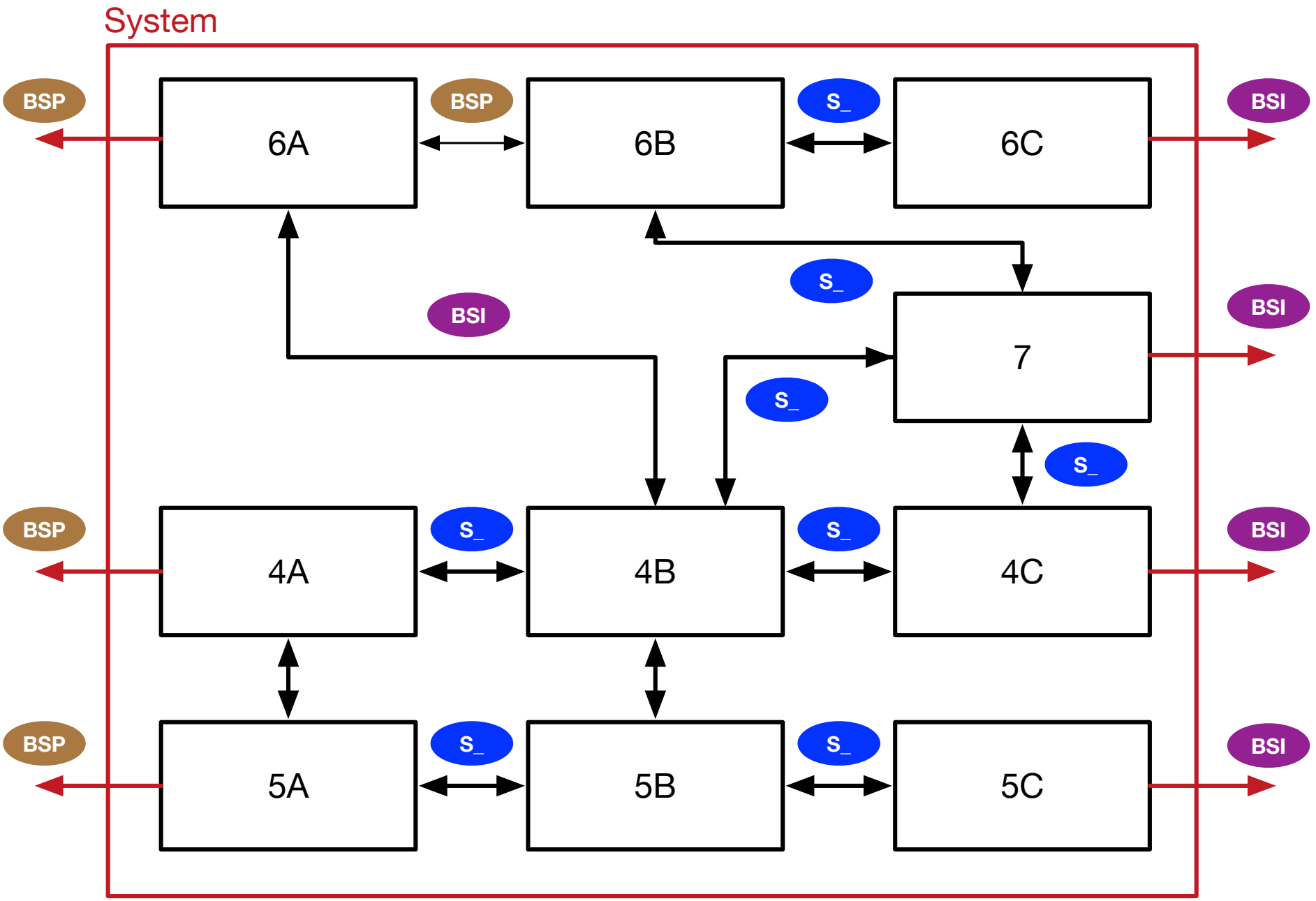
Goals of the presentation

- Decomposition
- Process
- GRAFITI input file
- GRAFITI parser and output files
- SMV model
- Problems

From



To


System
BSP
6A
BSP
6B
S_
6C
BSI
BSI
S_
7
BSI
S_
S_
BSP
4A
S_
4B
S_
4C
BSI
BSP
5A
S_
5B
S_
5C
BSI

Process

- Components, interfaces, and contracts (OCRA)
- Component models (NUXMV)
- **TRAIN2** modules (NUXMV)
- Verification (compositional, implementation, receptiveness, local)
- Traces visualization (SVG)

Example of contract:

$$\frac{R1 \rightarrow (A \wedge \neg B), R2 \rightarrow (B \wedge \neg A)}{\neg(R1 \wedge R2)}$$

GRAFITI - repo

```
<aw id="lt:10:AW:01AE" name="01AE"
    controlSiteNameId="lt:CSN:LT" interSiteNameId="
    lt:ISN:LT3" speedL="40" speedR="160">
  <neighbours>
    <neighbour side=".L" neighbourSide=".L"
        neighbourId="lt:10:AW:01BE" />
    <neighbour side=".P" neighbourSide=".G"
        neighbourId="lt:10:SRP:02E" />
    <neighbour side=".R" neighbourSide=".G"
        neighbourId="lt:10:TRDL:DE" />
  </neighbours>
```

GRAFITI - itinerary

```
<itinerary name="CE-162">
  <comp id="It:8_10:TCSN:015" position="1" inPort="
      .L" outPort=".SUBL" />
  <comp id="It:8_10:TCSN:015" position="2" inPort="
      .SUBR" outPort=".R" />
  <comp id="It:10:SRP:CE" position="3" inPort=".T"
     outPort=".G" />
  <comp id="It:10:SIGN:CE" position="4" inPort=".T"
      outPort=".G" />
  <comp id="It:10:SRP:01E" position="5" inPort=".T"
      outPort=".G" />
  <comp id="It:10:AW:01BE" position="6" inPort=".R"
      outPort=".P" />
  <comp id="It:10:SIGN:DXE" position="7" inPort=".G
     " outPort=".T" />
</itinerary>
```

Scala parser - GRAFITI

```
def load_AW(grNS: NodeSeq) = {
 (grNS \\ "aw").foreach { awINS =>
  val id = awINS.attribute("id").get.toString()
  val name = awINS.attribute("name").get.toString()
  val irId = (awINS\"inputAw"\"@irId").text

  val AWmap = Map("name" -> name, "speedL" ->
      speedL, "speedR" -> speedR,"irId" -> irId, "
      irSide" -> irSide, "vId" -> vId, "vSide" ->
      vSide, "lkgv" -> lkgv)
// Map icluding all the objects
   mapGR_Comp += id -> new AW(id, AWmap)
   val col = (awINS\\"graphicalInfo"\"@col").text
// Preparation of the SVG file
  svgF.addINFO(new svgAW(id,col,row,ori,sym,gtype))
 }
}
```

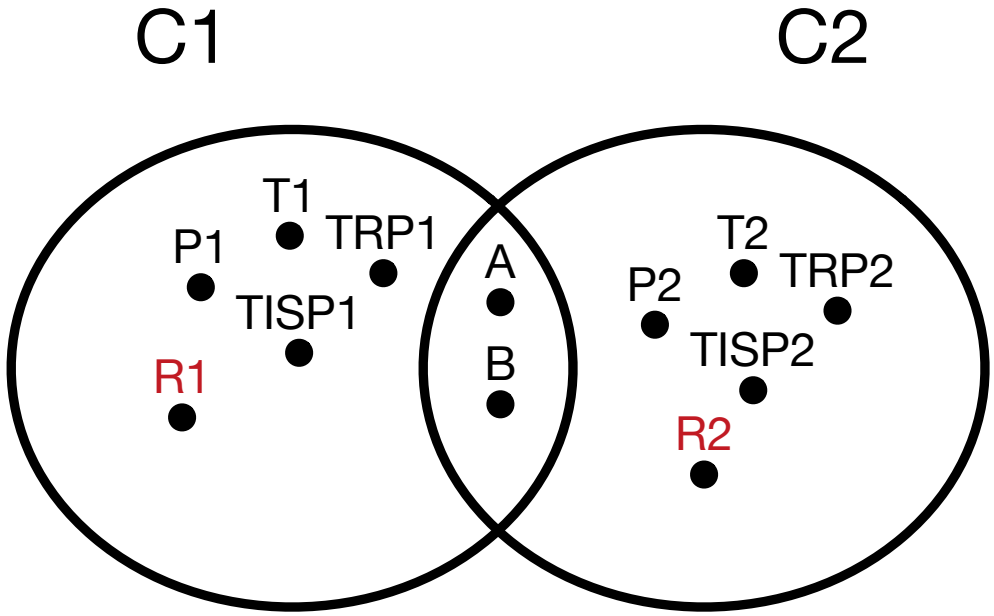
Scala parser - TRAIN2 generation

```
// from
var pF = mapTraPos.foldLeft(List[String]("beg")) {
    (u1, u2) => u1 ++ u2._2.PosiS.foldLeft(List[
    String]()) { (u3, u4) => u3 ++ List(u4.FROM)
    }}.map { oDir => "\t\tfront = " + oDir + " &
    next(moving) = " + rt.moving() }
val myCompPatterns.SIGxmlPattern(sN) = rt.retOriSig
    ().id
pF = pF.updated(0, pF(0) + " & " + sN + "open =
    TRUE ")
// to - adding the first transition
val pT = mapTraPos.foldLeft(List[String]()) { (u1,
    u2) => u1 ++ u2._2.PosiS.foldLeft(List[String
    ]()) { (u3, u4) => u3 ++ List(u4.TO) }} ++ List
    (": end")
```

NuXMV Model

```
#include "./theGenerics.smv"
#include "./train2_1.smv"

MODULE main
VAR
v211 : SSI21();
-- Inputs for OCRA              -- e.g. A
DEFINE
-- Outputs for OCRA component -- e.g. B

MODULE SSI21()
VAR
#include "./instancesRoutes.smv"
#include "./instancesU.smv"
#include "./instancesPoints"
#include "./instancesTC"
#include "./instancesTRP"
#include "./instancesTISP"

t1 : TRAIN2(self, TRUE);
t2 : TRAIN2(self, !t1.isStarting);
g1 : FrameAXioms(self);

ASSIGN
#include "./enaRou_NMH21_1.smv"
DEFINE
#include "./Pcond_NMH21_1.smv"
#include "./Rcond_NMH21_1.smv"
#include "./U_IRcond_NMH21_1.smv"
```

TBD



- Automatic isolation of components based on SSI parameters
- Prove concept is valid, NuXMV variables

$$C1 = \langle\ R_1, U_1, P_1, T_1, TRP_1, TISP_1, L_1, I_1, O_1 \rangle$$

$$C2 = \langle\ R_2, U_2, P_2, T_2, TRP_2, TISP_2, L_2, I_2, O_2 \rangle$$

$$C1 \cap C2 = I1 \cup O1 = I2 \cup O2 \text{ and } I1 = O2 \text{ and } I2 = O1$$

In previous formula:

$$A \in O_1 \text{ and A } \in I_2$$

$$B \in O_2 \text{ and A } \in I_1$$