Transformation Rules of Behavior Model from VxBPEL to fPromela Language

VxBPEL behavior specification is transformed to fPromela with the use of rules depicted in Table 1 where P1, P2,.., Pn depict BPEL specifications, fPromela-equvalent-P1, fPromela-equvalent-P2,…, fPromela-equvalent-Pn represent fPromela equivalent specifications of P1, P2,…, Pn .

Table 1.Transformation Rules

VxBPEL	fPromela	VxBPEL	fPromela
<partnerLink name = qname **.../>**	chan chan_qname	**<variable name** = qname **.../>**	byte qname
<receive partnerLink=qname **operation**=op_name **variable**=var1**.../>**	chan_qname!var1	**<reply partnerLink**=qname **operation**=op_name **variable**=var1 **.../>**	chan_qname?var1
<invoke partnerLink=qname **operation**=op_name **inputVariable**=var1 **outputVariable**=var2**.../>**	chan_qname!var1; chan_qname?var2;	**<assign>..** **<from>**var1**<from>** **<to>**var2**</to>..** **</assign>**	var2 = var1;
<flow> P1 P2 **</flow>**	fPromela-equvalent-P1; fPromela-equvalent-P2;	<**while**> <condition>expr</condition> P1 </**while**>	do :: expr -> fPromela-equvalent-P1 :: else -> break; od
<sequence> **P1** **P2** **</sequence>**	{fPromela-equvalent-P1}; {fPromela-equvalent-P2};	<**repeatUntil**> P1 <condition> expr </condition> </**repeatUntil**>	do :: fPromela-equvalent-P1 od unless {expr};
<if> <condition>expr1</condition> P1 **<elseif>** <condition>expr2</condition> P2 **</elseif>** **<else>** P3 **</else>** **</if>**	if :: expr1 → fPromela-equvalent-P1; :: expr2 → fPromela-equvalent-P2; :: else → fPromela-equvalent-P3; fi;	<**VariationPoint name**= qname> <**Variants**> <**Variant name**=varname1> <**VPBpelCode**> P1 </**VPBpelCode**></**Variant**> … </**Variants**> </**VariationPoint**>	/*For each variant exists in the variation point*/ gd :: varname1-> fPromela-equvalent-P1; :: else -> skip; dg;
<empty>	skip;		