15 Advanced SACM Capabilities with Examples (informative)

The following seven SACM Capabilities, 15.1-15.7, are built upon the materials in the previous sections with new items shown as well as additions to the existing items. "[Additions to xxxx]" are used to indicate the previous definitions of the item "xxxx" that the additions are applied to.

15.1 SACM Join

Joins can be used in SACM to support both Assurance Case patterns and various logical uses.

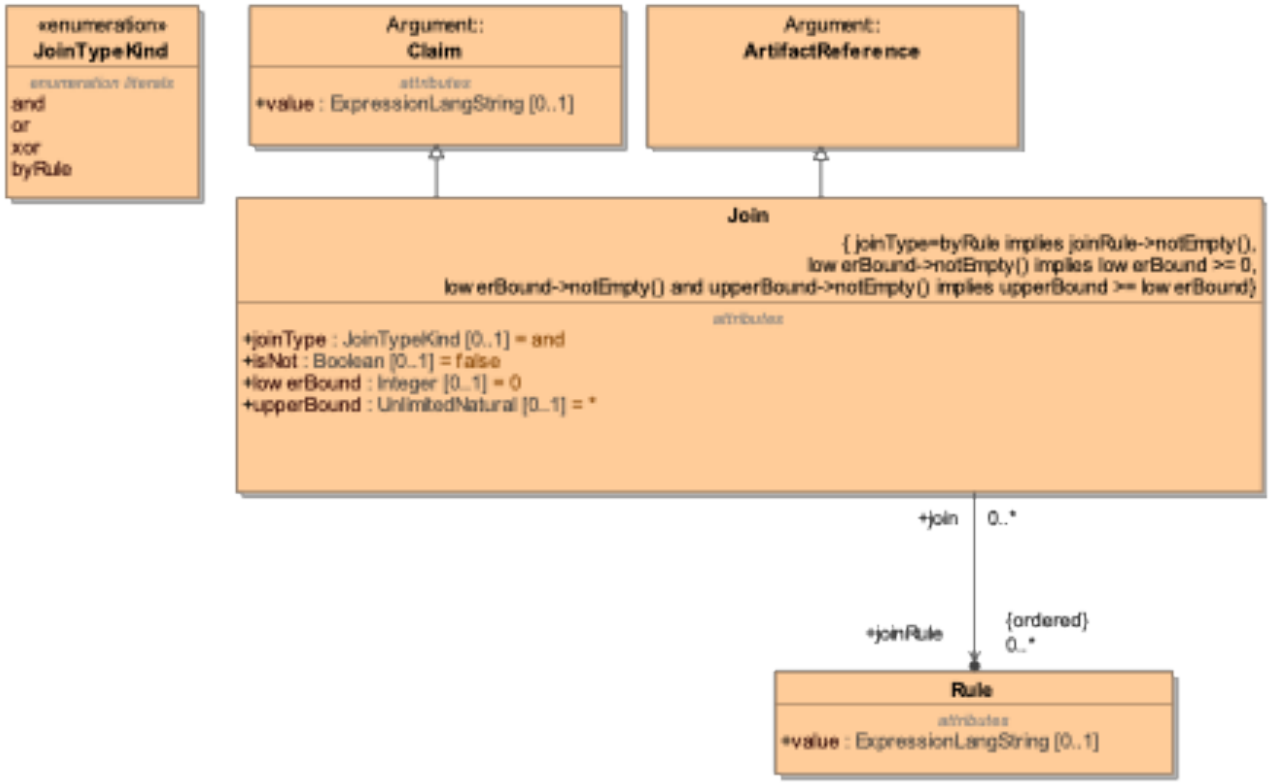

Figure 15.1.1 – SACM Join Diagram

15.1.1 Join

Join is a "logical" connector of multiple sources of relationships (whose targets end on the Join). For Patterns, the Join specifies a set of choices one can make when reifying this pattern. In some logics, this allows the logical collection (e.g. anding) of the sources of a relationship whose targets are this Join. The Join also can be a source of a relationship to some other final target SACMElement. This would be the equivalent of the original source to final target relationship with possible added semantics to the sources of those relationships.

Supertype

Argument::Claim, Argument::ArtifactReference

Attributes

joinType : joinTypeKind [0..1] = and - Defines the JoinTypeKind for this Join

isNot : Boolean [0..1] = false - Is the whole Join a Not (e.g. joinType=and and isNot=true, makes it a NAND operation).

lowerBound : Integer [0..1] = 0 - LowerBound for this Join, this is the least number of sources that can chosen for this Join (Join used for a pattern). This value is 0 or greater. If lowerBound is given and no upperBound is given, upperBound is assumed to be unlimited ("*").

upperBound : UnlimitedNatural [0..1] = * - UpperBound for this Join, this is the greatest number of sources that can be chosen for this Join (Join used for a pattern). If upperBound is given and no lowerBound, then assumed to be a single number specified by the upperBound (i.e. the LowerBound is equal to the UpperBound).

AssociationEnds

joinRule : Rule [0..*] {ordered} - if joinType=byRule, then this defines the Rule that joins the sources.

Constraints

UpperBoundGreaterThanOrEqualToLowerBound
If lowerBound and upperBound exist, then upperBound must be greater than or equal to lowerBound.
inv: lowerBound->notEmpty() and upperBound->notEmpty() implies upperBound >= lowerBound

LowerBoundGreaterThanOrEqualToZero
If lowerBound exists, then lowerBound is greater than or equal to zero.
inv: lowerBound->notEmpty() implies lowerBound >= 0

ByRuleMustHaveRule
If joinType=byRule, then joinRule must have a reference to a Rule.
inv: joinType=byRule implies joinRule->notEmpty()

15.1.2 JoinTypeKind

JoinTypeKind defines the relationships between the sources of the relationships that have targets that end on the Join.

EnumerationLiteral

and - "and" between the sources on the end of relationships

or - "or" between the sources on the end of relationships

xor - "xor" between the sources on the end of relationships

byRule - "byRule" defined relationship between the sources on the end of relationships

combine - is a combination of sources without any logical implications. These are treated as if the relationship that is sourced on the Join is coming directly to the sources.

15.1.3 Examples

The Join in a Template Example shows how a Join can be used in a pattern to define the various options allowed in the template.

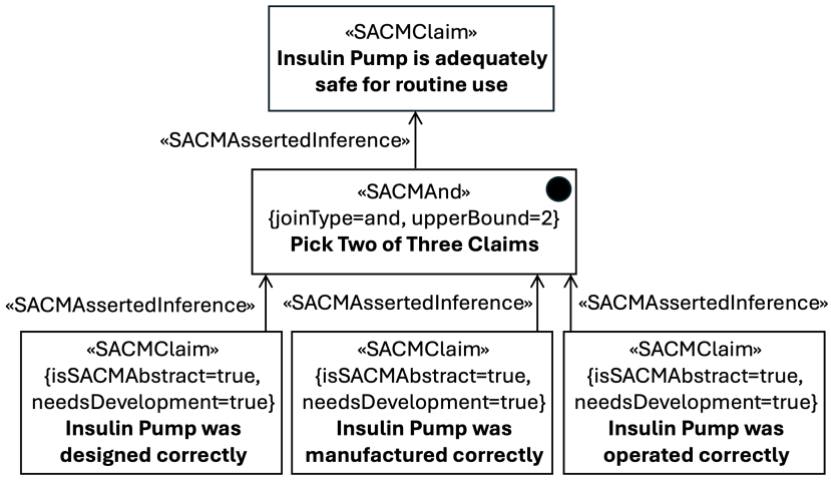

Figure 15.1.2.1 – Join in a Template Example

For a Deductive Argument, Claim B being true or Claim C being True (or Both), is enough to Specify that Claim A is true.

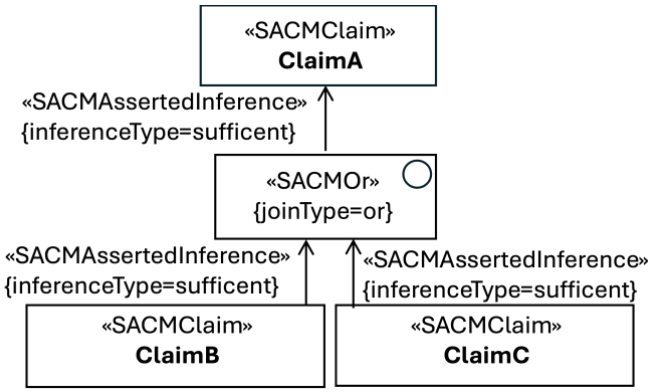

Figure 15.1.2.2 – Join in Deductive Logic Example

Join (in the profile), can be used in a structure like a Strategy (from GSN) or an Argument (from CAE). The Join (by either name or value tagged value) is the statement of the ArgumentReasoning,and required tagged values joinType=combine and a relationship with references to the AssertedRelationships.

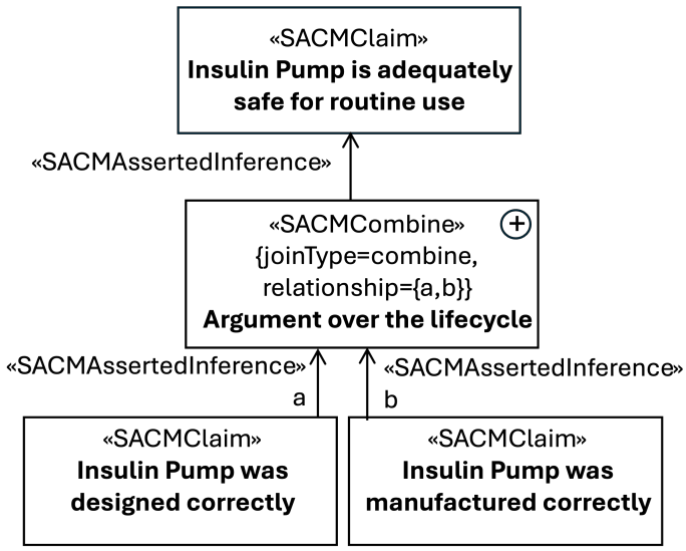

Figure 15.1.2.3 – Join as an ArgumentReasoning (Like a Strategy) Example