Structure of Data

Alphabetic Ordering of Types withing Groups

- **Symbolic Name in XML Configuration**: checkTypeDefOrderInGroup
- **Dependant Tags in XML Configuration**: -

This quality check analyzes type definitions within groups in the same module and throws a warning if type definitions within the same group are not alphabetically (or alpha-numerically, where numbers come before letters) ordered. The ordering is case-insensitive.

Grouping of Ports and Related Messages

- **Symbolic Name in XML Configuration**: checkPortMessageGrouping
- **Dependant Tags in XML Configuration**: -

This quality check verifies that port definitions are grouped together with all the message types or signatures referenced within the port definition. The ports and the message types / signatures related to them shall be grouped within the same group in the same module. Limited nested grouping is allowed, where the message / signatures may be grouped in a subgroup within the group in which the port to which they are related was defined. This can be best illustrated by examples:

```
module checkPortMessageGrouping {
    import from checkPortMessageGroupingExternal all;
    //correct examples

    group correct {
        type port port1 message {
            in m1;
            out m2, m3;
        }
        type port port2 message {
            inout m1, m3;
        }
        type port port3 message {
            inout m4, m3;
            out m1, m2;
            in integer, charstring;
            out integer;
        }
        type port port4 procedure {
            in sm1, sm2;
            out sm3, sm4;
```

```
        inout sm5, sm6;
    }
    type port port5 mixed {
        in sm1, m1;
        out integer;
    }
    group nested {
        group signatures {
            signature sm1();
            signature sm2();
            signature sm3();
            signature sm4();
            signature sm5();
            signature sm6();
        }

    }

    type integer m1;

    type record m2 {

    }

    type set m3 {

    }

    type record m4 {

    }

}
//incorrect examples

group incorrect {
    type port port6 message {
        in m1, m2;
        out m3;
    }
    type port port7 message {
        inout checkPortMessageGroupingExternal.m5;
        out m6;
    }
    type port port8 procedure {
        inout sm1, sm2;
        out sm3;
    }
```

```
    }

    type record m6 {

    }

    type port port9 message {
        inout m6, m7;
    }

    group g1 {
        type set m7 {

        }
        type record m8 {

        }
    }

}
```

In the above module definition, the contents of group `correct` are OK, where as the contents of group `incorrect` are not, because the message types are not defined under the same group where the port they are related to is defined.

If a port is defined outside a group, it automatically means that the related message types cannot be defined within the same group and they are not further analyzed. If a message type is defined outside a group, it also means that it is not in the same group as the port definition it is related to. If a message type is defined within a group with the same name as the one where the port it is related to is defined, but within a different module, it is also considered a violation of the constraint and a warning will be thrown. If a message type related to a port type definition cannot be resolved, it is considered a violation of the constraint as well and an appropriate warning is thrown.

The standard (first) line numbers in the output indicate the reference point - on which lines within the port type definition has the violating message type been referenced. Additionally, next to the message type / signature name and the port type name, location triples <startLine,moduleName,groupName> are provided to facilitate the easier identification and localization of the elements violating the constraint.

Additionally, if nesting is present as in the above example, but does not violate the constraints, an information message will be provided in the output to inform the user of the occurrence.

No All Keyword in Port Type Definitions

- **Symbolic Name in XML Configuration**: checkNoAllKeywordInPortDefinitions
- **Dependant Tags in XML Configuration**: -

The check makes sure that there are no all keywords in port type definitions.