C2DF

> **Legacy Project**
>
> KAOM is not maintained anymore and hence not part of any KIELER release.

Project Overview

Responsible:

- Miro Spönemann
- Sven Gundlach

Introduction

Synthesis of data flow diagrams from annotated C programs (C2DF) is a command line program for the automated generation of data flow diagrams. It use KAOM for generation and the automatic layout provided by KIML.

Manual

To work with C2DF you must pass as the first source and a destination directory as an argument. C2DF generates then in the target directory for each annotated source file from the source directory or a subdirectory thereof a *.KAOT*-file. Also, a subdirectory called *xml* will be created in the target directory. This have to be deleted.

General description of annotations:

To declare annotations, you need to put in a C-comment before the annotated object. This comment will be initiated by /* and completed with */. Before the start of each content must be written the keyword *@kaom*. Then follow the individual contents.

IDs must consist only of a particular character set. These are all signs of the Latin alphabet in uppercase and lowercase letters (*a-z,A-Z*), as well as Hindu-Arabic numerals in European notation(*0-9*). In addition, spaces and underscores are allowed. Other characters are replaced internally by underscores _. This also applies to the diacritical marks *ä,ö,ü* , and the ligature *ß*. Similarly,it is not allowed to write the Keywords *input:, output:*, *link:*, *content:* and *toplevel:* in IDs. According to the ASCII are the characters 32, 48-57, 65-90, 95 and 97-122 allowed in ids.

Labels can hold all characters, except quotation marks, commas, semicolons, at sign, asterisks with a backslash and backslashes with a asterisk.

The object names must be identical with the object names in the source code.

There is no check if IDs are used or defined.

example of an full annotation

```
/*!  @kaom
input:    ID_1 "label", ID_2 "label", ... ;
output:   ID_1 "label", ID_2 "label", ... ;
link:     source -> context.target, context.source -> context.target, source -> target, ... ;
content:  function_1:ID "label", function_2:ID "label", ... ;
toplevel: ID "label";
*/
```

Specification of inputs:

The specification of the inputs of an actor must be initiated by the keyword *input:*. Then follows the list of inputs. Each entry must have a unique ID in the actor. Optional behind the ID can be any label in quotation marks. Other entries are separated with commas. The list of all inputs must be terminated with a semicolon.

example for specifying inputs

```
input: ID_1 "label", ID_2 "label", ... ;
```

Specification of outputs:

The specification of the outputs of an actor must be initiated by the keyword *output:*. Then follows the list of outputs. Each entry must have a unique ID in the actor. Optional behind the ID can be any label in quotation marks. Other entries are separated with commas. The list of all outputs must be terminated with a semicolon.

example for specifying outputs

```
output: ID_1 "label", ID_2 "label", ... ;
```

Specification of data flows:

The specification of the data flows of an actor must be initiated by the keyword *link:*. Then follows the list of the data flows. The ID of the source and the ID of the target must be specified for each entry. Source and destination are separated by a right-leaning arrow indicating the direction of data flow. If source or destination are defined in another actor as the data flow, then the ID of the actor of the source or destination must be specified. The ID of the actor is placed directly in front of the ID of the source or target, separated by a dot. Actors, in which source and destination are defined, may not differ by more than one hierarchical level. Other entries are separated with commas. The list of all data flows must be terminated with a semicolon.

example for specifying data flows

```
link: source -> context.target, context.source -> context.target, source -> target, ... ;
```

Specification of included actors:

The specification of the included actors of an actor must be initiated by the keyword *content:*. Followed by the list of the included actors. Each entry must be specified with the name of its represented function in the source code. Followed by a unique ID within the actor, which is separated from the function name with a colon. Optional behind the ID can be any label in quotation marks. Other entries are separated with commas. The list of all included actors must be terminated with a semicolon.

example for specifying included actors

```
content: function_1:ID "label", function_2:ID "label", ... ;
```

Specifying top-level actors:

The specification of the top-level actors of a data flow diagram must be initiated by the keyword *toplevel:*. The specification must be placed in the annotation of the represented function. The entry consists of an unique ID in the actor. Optional behind the ID can be any label in quotation marks. The entry must be terminated with a semicolon.

example for specifying top-level actors

```
toplevel:ID "label";
```

Special case:

It is possible to force data flows in contrary to the general specifications . This is particularly the case when there are multiple top-level actors. The specification of data flows between these actors is set to the specifications of other data flows. For the specification, all predecessors of the actor and the actor itself, which define the source and target, must be specified by IDs. For the specification the path to the source and to the destination must be specified. Both are separated by a right-leaning arrow. A path consists of the ID of the source (or destination) and the ID of the actor, in which it is defined, and the IDs of all textual comprehensive actors. The ID of the defining actors is placed in front of the ID of the source (or destination) and separated by a colon. The IDs of the textual comprehensive actors, are arranged in descending order, placed in front of the ID of the defining actor. The IDs of all actors in the path are separated from each other by an underscore . Source and destination can differ in more than one hierarchical level. All characters that are replaced with underscores in an ID must also be replaced with underscores in the specifying.

example for specifying a special data flow

```
link: source -> ID0_ID1_ID2_..._ID:target, ... ;
```