Let an event's composite definition be $e \lhd d$, and the complex event e^1 appear in d with $e^1 \lhd d^1$. Then the complex event e is said to be *refinement-connected* (or *r-connected* for short) to the definition d'. Similarly e is said to be r-connected to d^n if there exists an acyclic chain of r-connected relations where $e \lhd d$, with $e^1 \in A^c_d$ and $e^1 \lhd d^1$, and with $e^2 \in A^c_{d^1}$ and $e^2 \lhd d^2$, ... with $e^n \in A^c_{d^{n-1}}$ and $e^n \lhd d^n$. If $A^c = \emptyset$ in a given composite definition d, it is said to be *irreducible*.

Given $e \lhd d^1$ and an r-connected definition d^2, then its flattened definition is $d^{1'} = \langle A^p_{d^{1'}}, A^c_{d^{1'}}, KL_{d^1}, L_{d^{1'}}, \preceq_{d^{1'}}, \lambda_{d^{1'}} \rangle$ of e where $A^p_{d^{1'}} = A^p_{d^1} \cup A^p_{d^2}$, $A^c_{d^{1'}} = A^c_{d^1} \cup A^c_{d^2}$, $KL_{d^{1'}} = KL_{d^1} \cup KL_{d^2}$

The generation of an event's flattened definition refinements involves expanding its set of complex events, primitive events, context relation literals, partial orders and labels with those appearing its r-connected definitions, and the addition of a set of partial order relation over the maximal labels of the re-connected definition and the event it defines. See Algorithm We assume that every complex event in an event's composite definition (and its r-connected definition) is r-connected to an irreducible composite definition. We further consider that a complex definition is associated with a single complex event, and that each label in definition is unique with respect to the set of labels appearing in definitions $\mathcal{D}$.

Algorithm 1 generates a set of partial orders $\Phi = \{d_1, ..., d_2\}$, associated with a complex events from an event's composite definition $e \lhd d$, taking into account all possible partial orderings over the maximal labels of its partial order $\preceq_d$. Each element d_i comprises all the primitive events, complex events and context relation literals that appear in every r-connected definition.

Algorithm 1 *Generate flat refinements set for complex events*

e: complex event
$\mathcal{D}$: Set of composite definitions
$\lhd$: Set of composite definitions relations
$\Phi = \{d_1, ..., d_n\}$

function GENERATE_FLAT_REFINEMENTS(e)
    $d \leftarrow$ *find_complex_definition*(e,$\lhd$,$\mathcal{D}$)
    ML $\leftarrow$ *find_maximals*($\preceq_d, L_d$)
    $\Phi = \emptyset$
    for all $l \in ML$ **do**
        $(v, FD) \leftarrow$ *compress_definition(e, l)*
        $v.\lambda = v.\lambda \cup \{e\}$
        for all $f \in FD$ **do**
            $\Phi = \Phi \cup \langle v.\mathcal{A}^p, v.\mathcal{A}^c, v.KL, v.L, v.\preceq \cup f, v.\lambda \rangle$
        end for
    end for
    return Φ
end function

Algorithm 2 outlines a recursive function that returns for a given complex event e and label l, (a) an composite definition whose complex events, primitive events, context relation literals, assignments and labels are expanded with those appearing in its r-connected definitions, and (b) a set of partial orderings over the maximal labels appearing in its definition and l. The function *find_complex_definition* returns a single definition d such that $e \lhd d$ and $d \in \mathcal{D}$.

Algorithm 2 *Compress definitions for complex events*

1: e: complex event
2: l: label
3: $\mathcal{D}$: Set of composite definitions
4: $\lhd$: Set of composite definitions relations
5:
6: **function** COMPRESS_DEFINITION(e, l)
7:     $d \leftarrow$ *find_complex_definition*(e,$\lhd$,$\mathcal{D}$)
8:     **if** *irreducible*(d) **then**
9:         *ML* $\leftarrow$ *find_maximals*($\preceq_d, L_d$)
10:         *FD* $\leftarrow$ *fix_order*(*ML*, l)
11:         **return** (d, FD)
12:     **else**
13:         $d' = d$
14:         *poe*$= \emptyset$
15:         **for all** $nl \in L_{d'}$ **do**
16:             **for all** $c \in \lambda_{d'}(l)$ **do**
17:                 **if** $c \in A^c_{d'}$ **then**
18:                     $(v, FD) \leftarrow$ *compress_definition(c, nl)*
19:                     *poe* $\leftarrow$ *update*(*poe*$, \preceq_{d'}, \preceq_v, FD$)
20:                     $d'.\lambda = d'.\lambda \cup v.\lambda$
21:                     $d'.L = d'.L \cup v.L$
22:                     $d'.A^p = d'.A^p \cup v.A^p$
23:                     $d'.KL = d'.KL \cup v.KL$
24:                     $d'.A^c = d'.A^c \cup v.A^c$
25:                 **end if**
26:             **end for**
27:         **end for**
28:         *ML* $\leftarrow$ *find_maximals*($\preceq_{d'}, L_{d'}$)
29:         *FD* $\leftarrow$ *fix_order*(*ML*, l)
30:         *poe* $\leftarrow$*create_compressed_sets*(*FD*,*poe*)
31:         **return** (d', *poe*)
32:     **end if**
33: **end function**

The function *find_maximals* returns, for a given set of labels and partial order over these, the set of maximal labels. The function *fix_order* creates a set of partial order sets $\{\phi_i\}$, one for each maximal label l' in *ML* such that for all $l_j \in ML$ where $l_j \neq l'$, $l_j \preceq l' \in \phi_i$ and where $l \notin ML$, it includes $l' = l \in \phi_i$. The function *update* on line 19 updates the set of partial order sets, appending each element with $\phi_i \in FD$ and the partial orders $\preceq_{d'}$ and $\preceq_v$. *create_compressed_sets* produces a new set of partial orders by considering each of the partial orders over the maximal labels of the current definition and those in *poe*. The final number of elements Φ is bound by the product of the number of maximal labels in each r-connected definitions.