

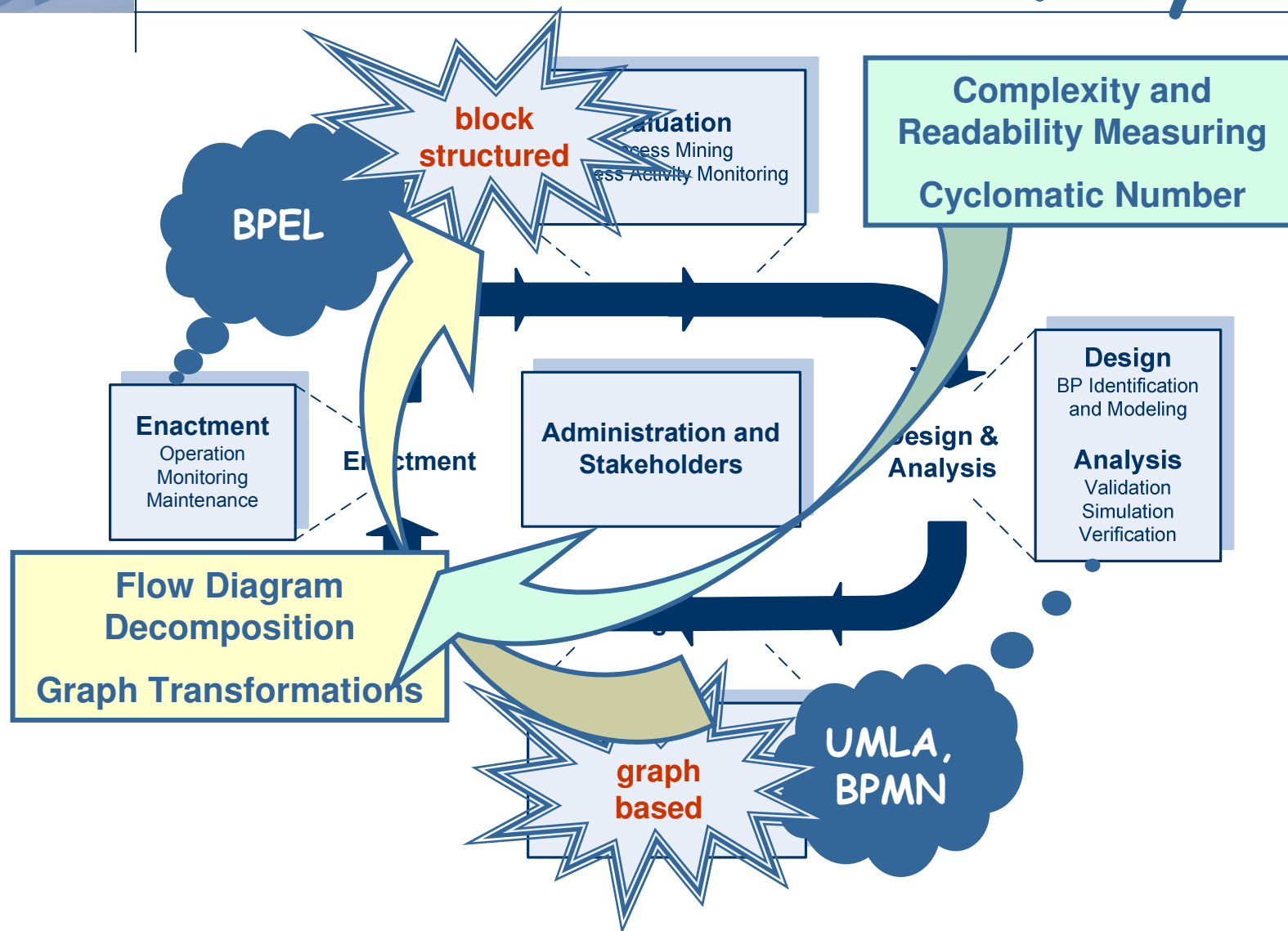
Towards Model Structuring Based on Flow Diagram Decomposition

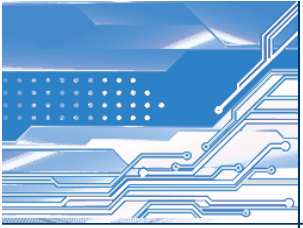
Arend Rensink, Maria Zimakova

*The First Workshop on Behavioural Modelling
in Model-Driven Architecture (BM-MDA)*

23 June 2009

Business Process Lifecycle



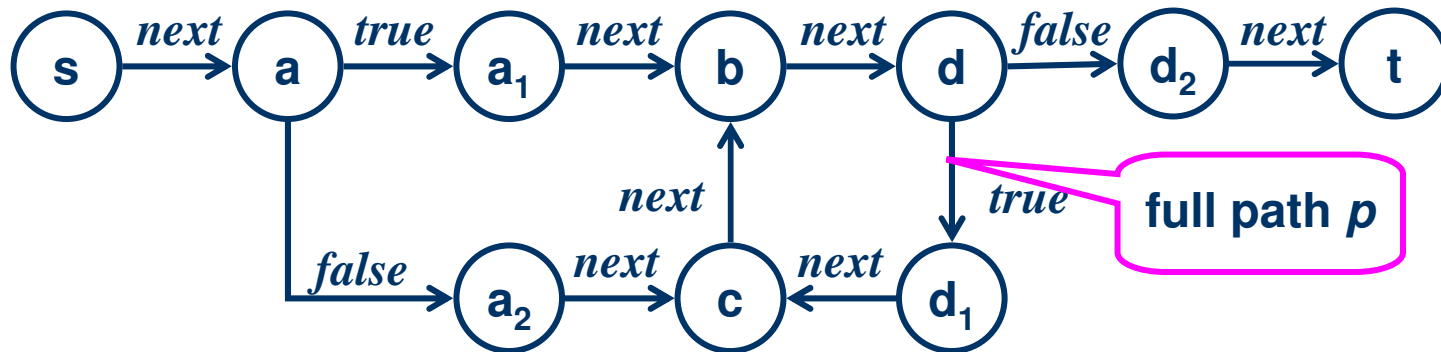


Outline

- Graphs and flow graphs
 - main flow graph definitions
 - strong and weak decomposition
- Flow diagram decomposition
 - base subdiagrams
 - SCC decomposition
 - complexity measuring
- Implementation
 - GROOVE implementation



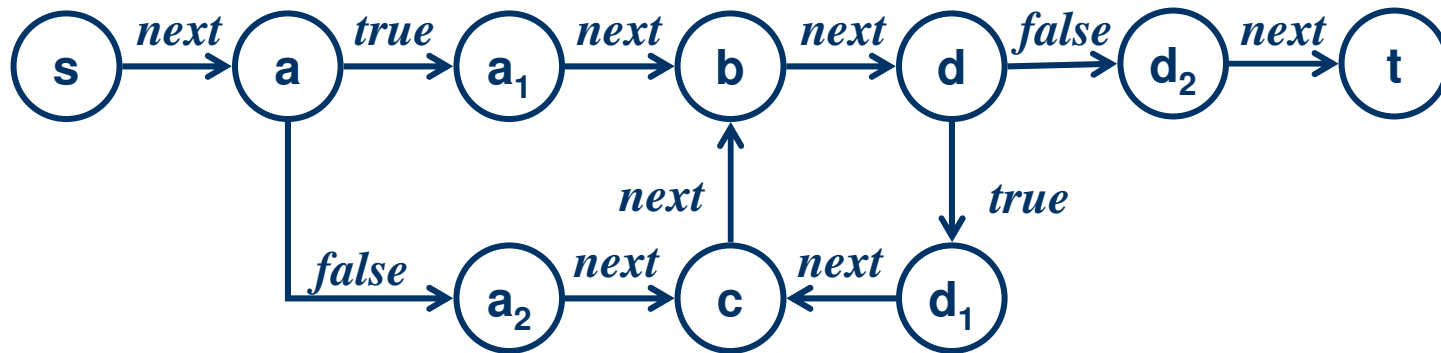
Flow Graph Example



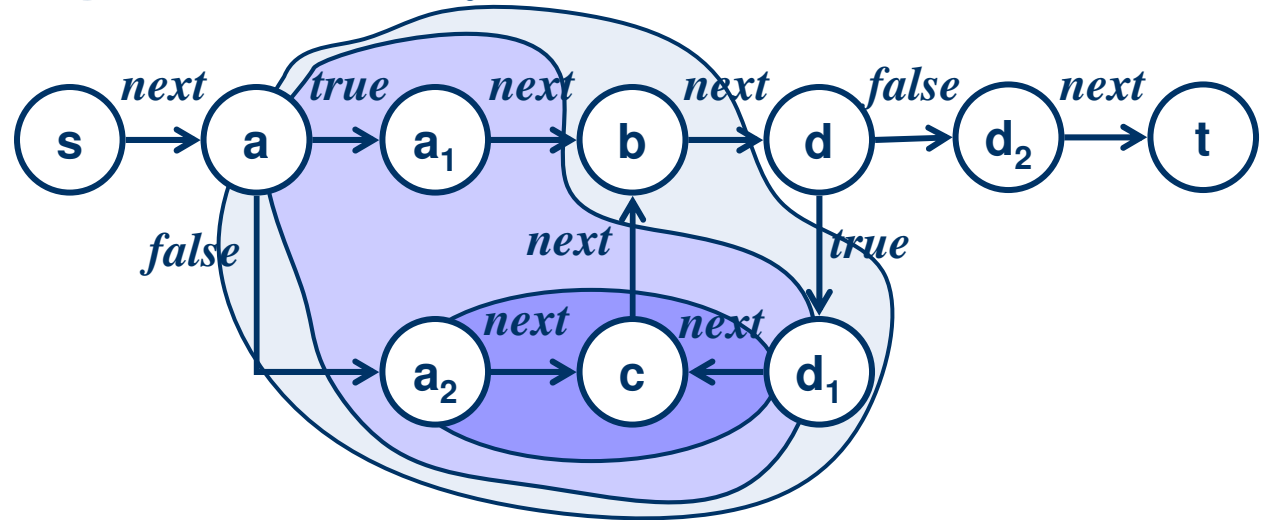
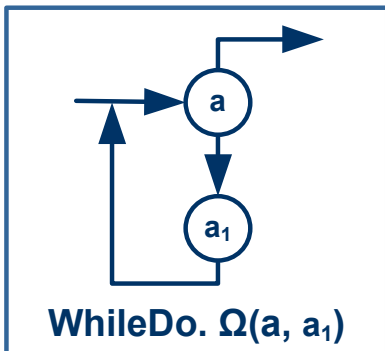
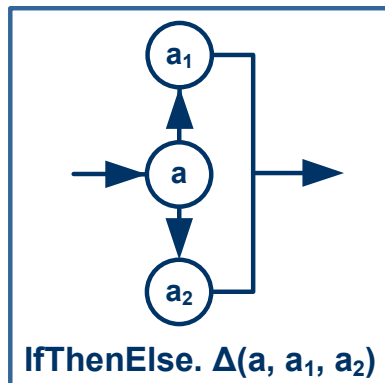
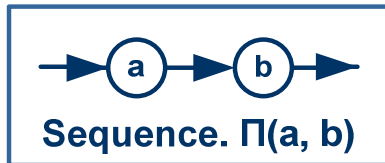
$Seq(p) = (s \text{ next } a \text{ true } a_1 \text{ next } b \text{ next } d \text{ true } d_1 \text{ next } c \text{ next } b \text{ next } d \text{ false } d_2 \text{ next } t)$

- **Full path p** in the flow graph is a path from the start node s to the terminal node t .
- **Execution sequence** $Seq(p)$ is the word representation of the full path p .

Flow Graph Example



Flow Diagrams. Strong Decomposition



Definition 1. A flow diagram $\Phi = (G, s, t)$ is *strongly decomposable* (or *well-formed*) if there exists an expression exp in the Sig-algebra FlowGraph such that

$$\text{FlowGraph}[[exp]] \cong \Phi.$$



Flow Diagrams. Strong Decomposition

□ Sig =

Sort: $fg, pred, func$;

Oper: $empty, elem, \Pi, \Delta, \Omega$;

par: $empty \rightarrow fg$,

$elem: func \rightarrow fg$,

$\Pi: fg\ fg \rightarrow fg$,

$\Delta: pred\ fg\ fg \rightarrow fg$,

$\Omega: pred\ fg \rightarrow fg$.

□ FlowGraph:

$$D_{fg} = \Theta,$$

$$D_{func} = \theta_{func},$$

$$D_{pred} = \theta_{pred},$$

$$f_{empty} = \varepsilon \in \Theta,$$

$$f_{\Pi} : D_{fg} \times D_{fg} \rightarrow D_{fg},$$

$$(\Phi_a, \Phi_b) \mapsto \Pi[\Phi_a / v_a][\Phi_b / v_b]$$

$$f_{\Delta} : D_{pred} \times D_{fg} \times D_{fg} \rightarrow D_{fg},$$

$$(\alpha, \Phi_a, \Phi_b) \mapsto \Delta[\Phi_a / v_a][\Phi_b / v_b]$$

$$f_{\Omega} : D_{pred} \times D_{fg} \rightarrow D_{fg},$$

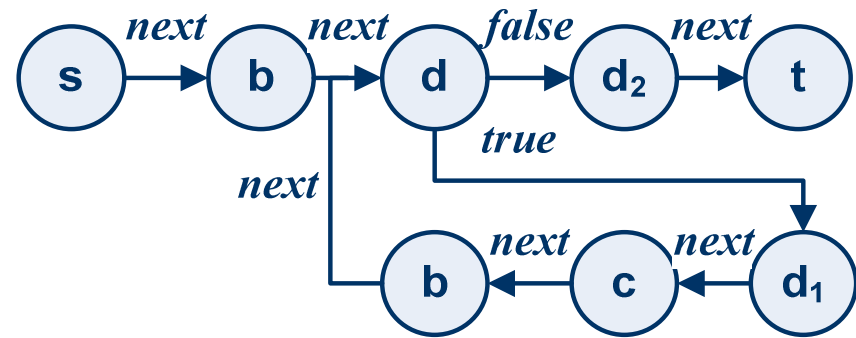
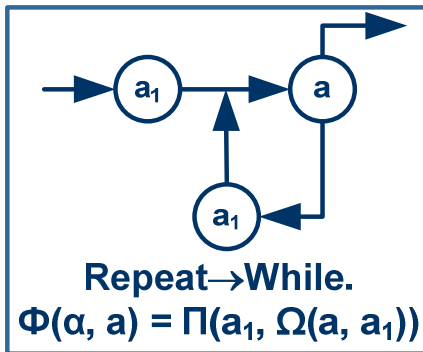
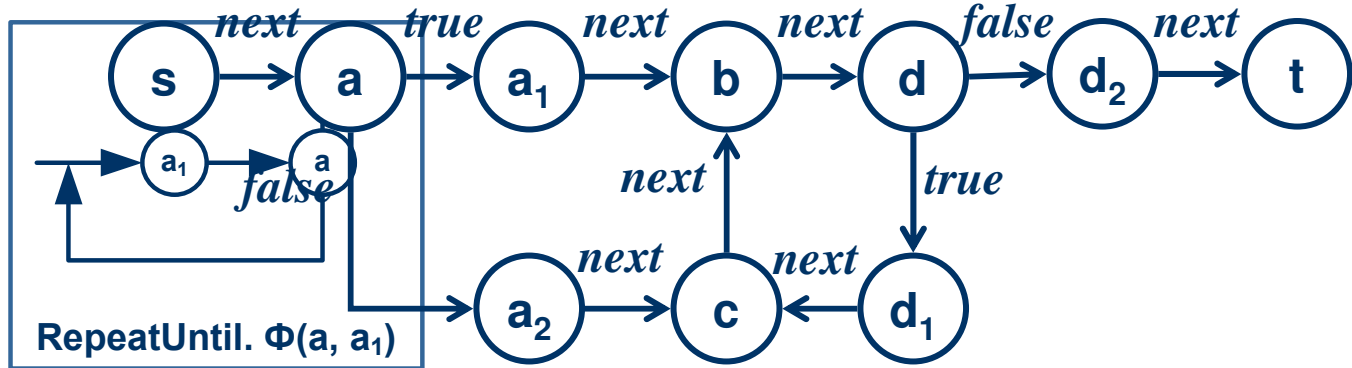
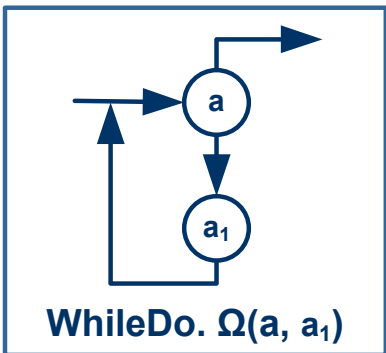
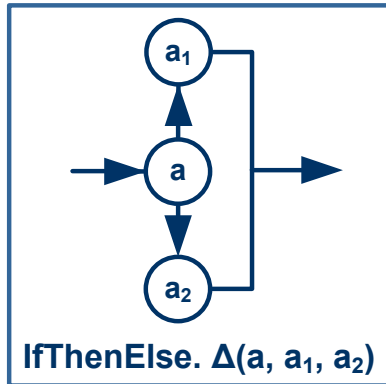
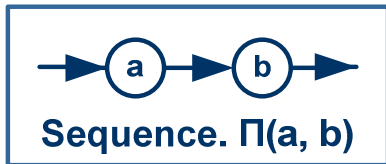
$$(\alpha, \Phi_a) \mapsto \Omega[\Phi_a / v_a].$$

Definition 1. A flow diagram $\Phi = (G, s, t)$ is *strongly decomposable* (or *well-formed*) if there exists an expression exp in the Sig-algebra FlowGraph such that

$$\mathbf{FlowGraph}[[exp]] \cong \Phi.$$

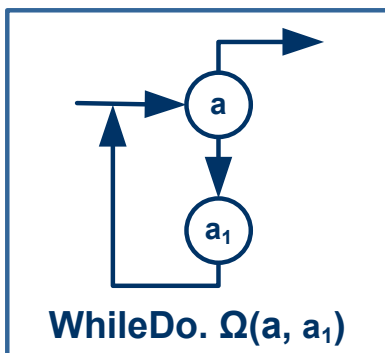
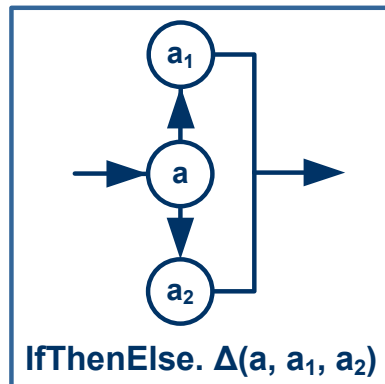
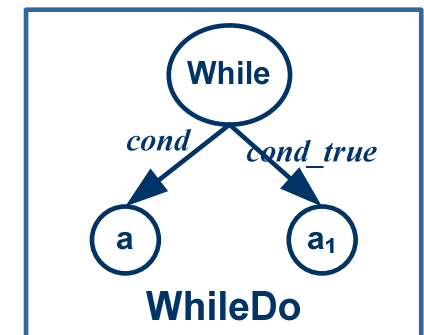
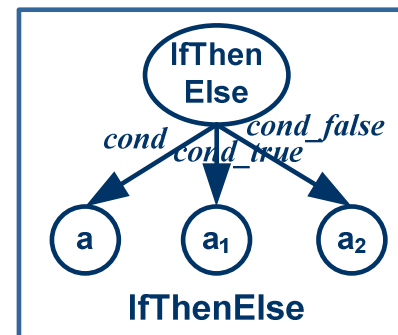
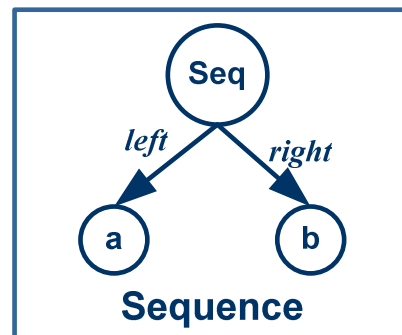
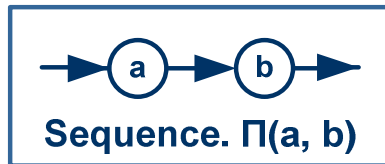


Flow Diagrams. Weak Decomposition



Definition 3. A flow graph Φ is *weakly decomposable* if $\Phi \sim \Phi'$ for some strongly decomposable flow graph Φ' .

Syntax Tree Decomposition



Definition 4. A *syntax tree decomposition* of a weakly decomposable flow graph $\Phi = (G, s, t)$ is a following morphism:

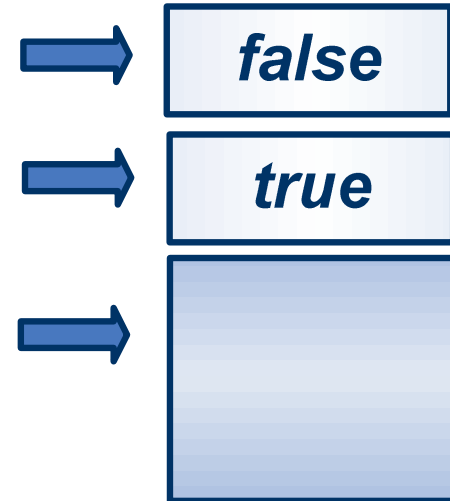
$$STD: \Phi \mapsto \{ \text{SyntaxTree}[[exp]] \mid \text{FlowGraph}[[exp]] \cong \Phi' \sim \Phi \}$$

where $\Phi' = (G', s, t)$ is an strongly decomposable flow graph equivalent to Φ .



Base Subdiagram Decomposition (Böhm, Jacopini)

- ❑ Three new functions T, F, K
- ❑ One new predicate ω
 - ❑ ω is *true* iff the last boolean variable value is *true*



Theorem 1. For any flow graph Φ_1 there exists (at least) one *equivalent strongly decomposable* flow graph Φ_2 extended by the functions K, T, F and predicate ω ; in other words, ***any flow graph is weakly decomposable.***

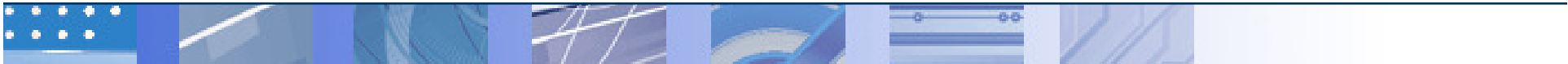




Base Subdiagram Decomposition. Extended Equivalence

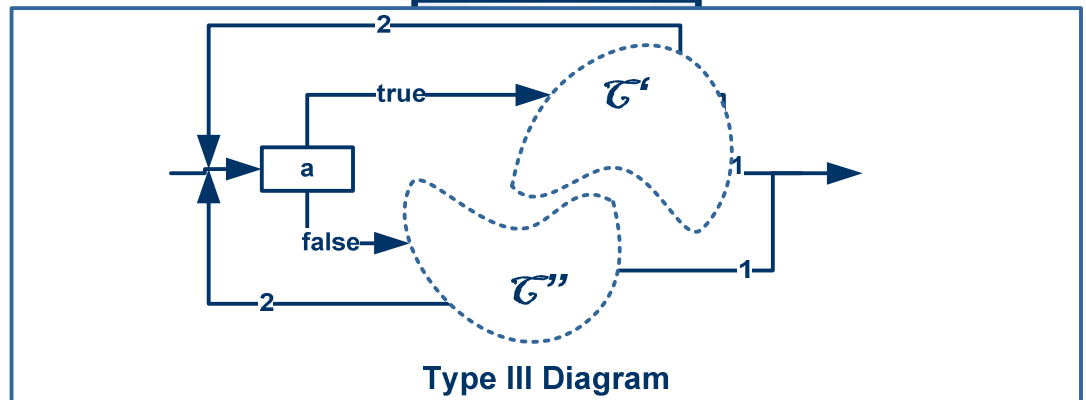
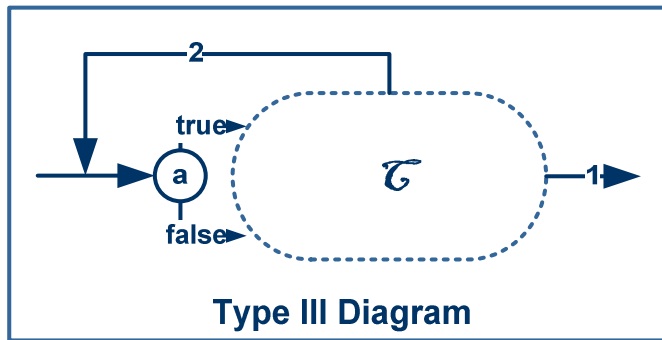
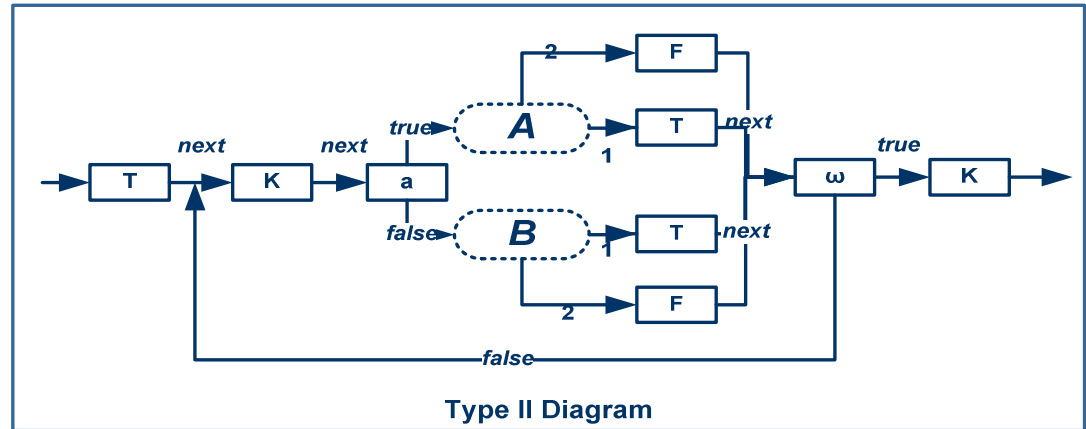
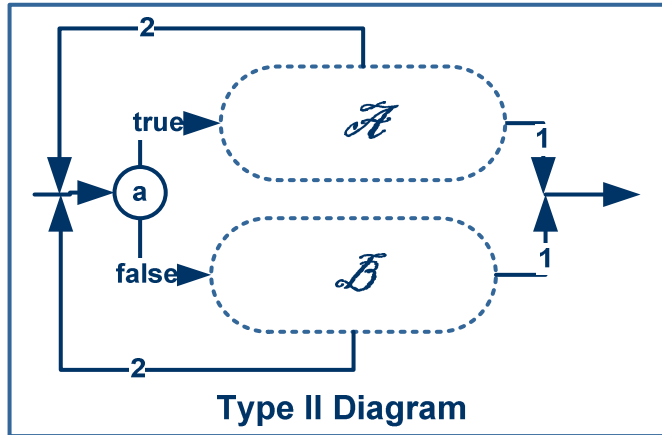
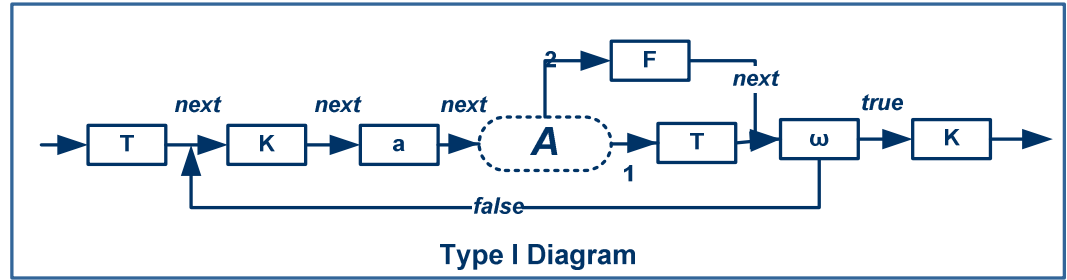
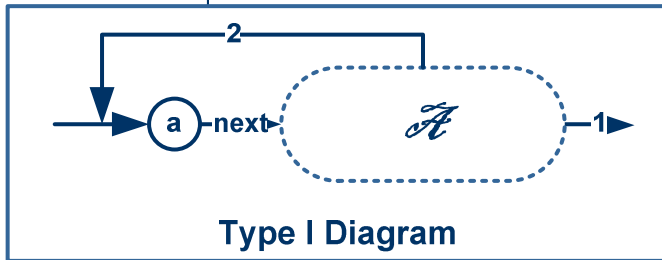


$$\text{Sat}(\mathbf{w}) = \begin{cases} \varepsilon & \text{if } \exists i, j \in [2, n-2], i < j : \\ & x_i \in \{T, F\}; x_j = \omega; \\ & x_{j+1} \in \{true, false\} \setminus \{\tilde{x}_i\} \text{ and} \\ & \forall k \in [i+1, j-1]: x_k \notin \{T, F, K, \omega\}; \\ \mathbf{w} & \text{otherwise} \end{cases}$$

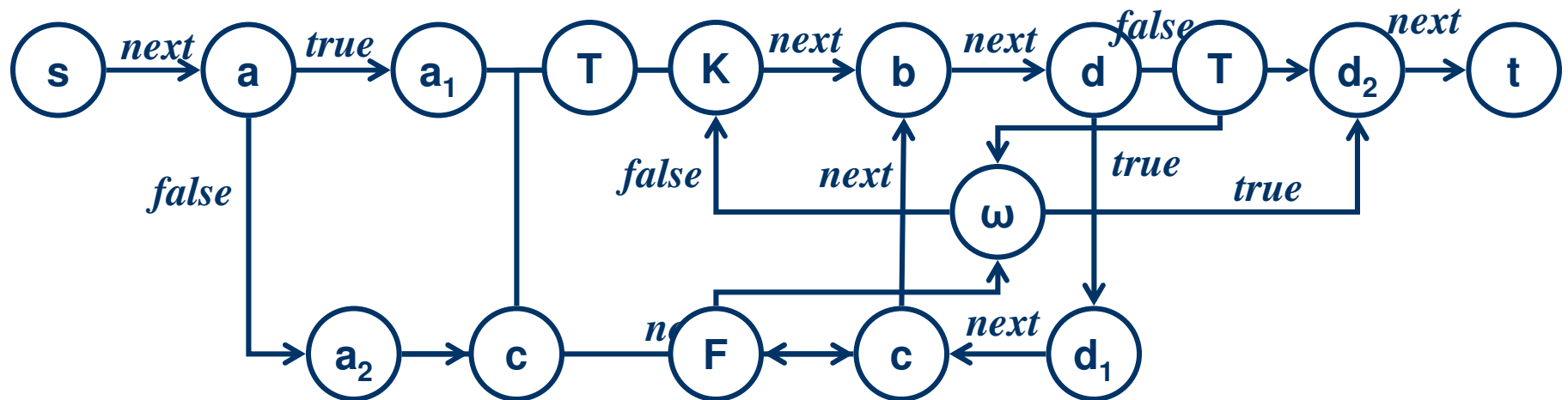




Base Subdiagram Decomposition. Normalization

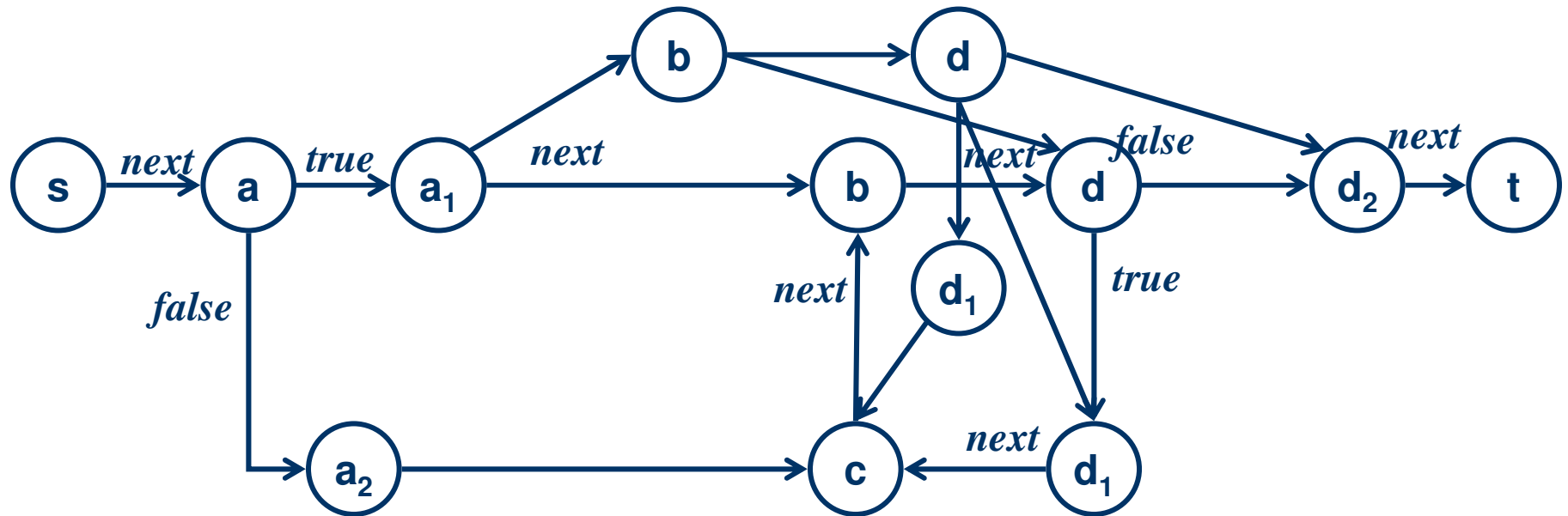


SCC Decomposition (Peterson)





SCC Decomposition-2





Complexity Measuring

Definition 1 (cyclomatic number) [McCabe]

The cyclomatic number $v(G)$ of a flow graph Φ with π predicates is $v(\Phi) = \pi + 1$.

Definition 2 (essential complexity)

The *essential complexity* $v_e(\Phi)$ is used to reflect the lack of structure: $v_e(\Phi) = v(\Phi) - m$, where m is the number of base subdiagrams Ω or Δ .

Definition 3 (full complexity)

Let $v_d(\Phi)$ - the number of duplicated nodes in a graph Φ . Then the following definition of *full complexity* $V(\Phi)$ is used: $V(\Phi) = [v(\Phi) + v_d(\Phi)] \times v_e(\Phi)$.



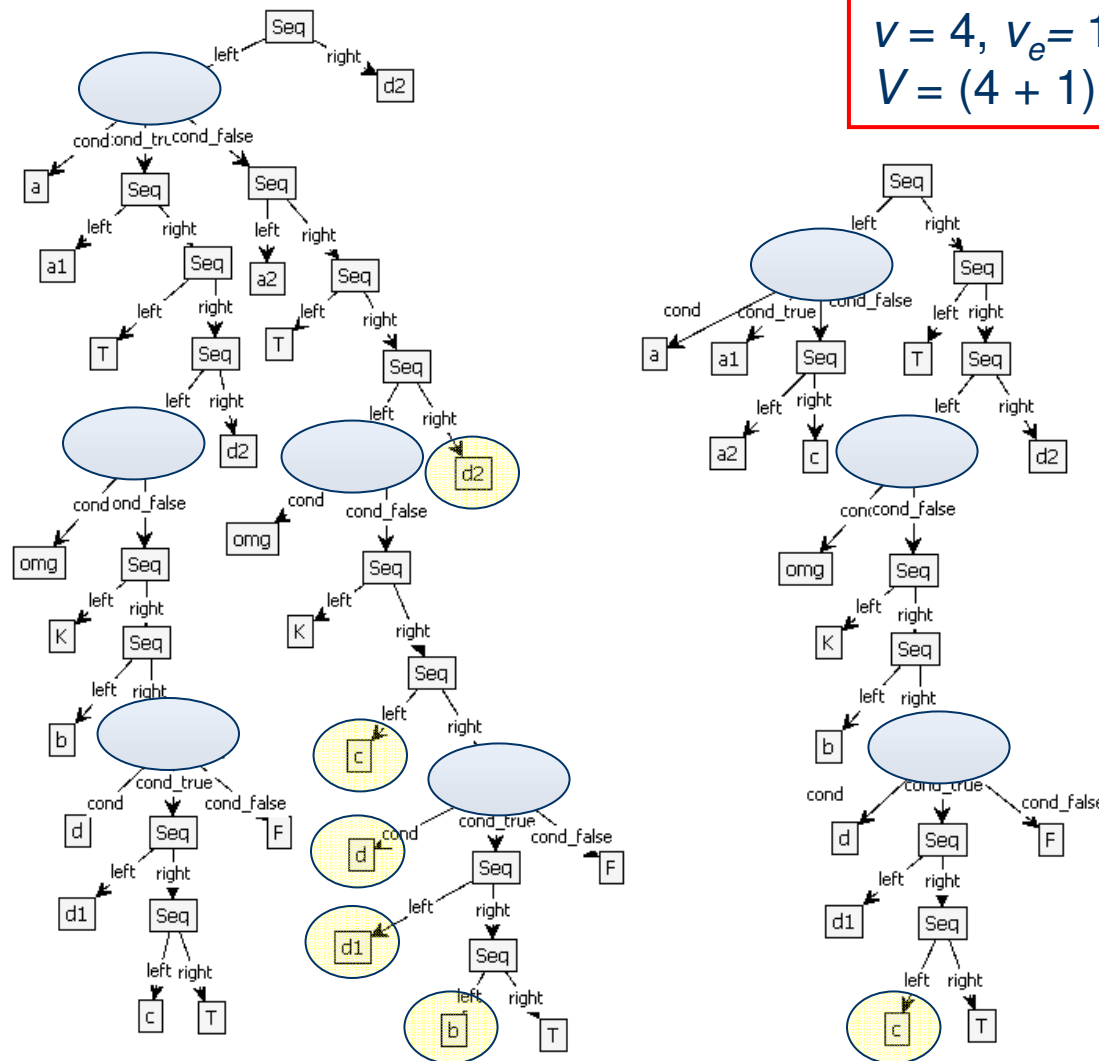
GROOVE Implementation

$$v = 3, v_e = 3, v_d = 0, \\ V = (3 + 0) * 3 = 9$$

$$v = 6, v_e = 1, v_d = 5, \\ V = (6 + 5) * 1 = 11$$

- ❑ Copy syntax tree
- ❑ Decomposition
- ❑ Contraction
- ❑ SCC selection
- ❑ Get syntax tree

$$v = 4, v_e = 1, v_d = 1, \\ V = (4 + 1) * 1 = 5$$





Results

#	Initial flow graph		Böhm-Jacopini method (deterministic)		SCC method (non-deterministic)				
	<i>n</i>	<i>V</i>	<i>n</i>	<i>V</i>	Result count	Min <i>V</i>		Max <i>V</i>	
						<i>n</i>	<i>V</i>	<i>n</i>	<i>V</i>
1	8	3	8	3	1	8	3	8	3
2	9	9	12	4	1	12	4	12	4
3	10	9	26	11	5	17	5	32	12
4	14	36	38	18	12	25	11	63	29
5	50	156	82	64	52	71	32	82	64
6	100	276	237	154	72	112	84	289	312





Conclusion

- Implementation of flow graph decomposition
 - Base subdiagram decomposition
 - SCC decomposition
- Complexity measuring
 - Evaluation of decomposition methods
 - Readability degree
- Future work
 - Flow graphs with parallelism
 - Implementation with real BP models





**Thank you for
your attention!**

