An Analysis and Taxonomy of Unstructured Workflows

Rong Liu and Akhil Kumar
Smeal College of Business
The Pennsylvania State University
University Park, PA 16802, USA
{rongliu, akhilkumar}@psu.edu

Outline

- Motivation
- Objectives
- Introduction
  - Workflow definition, structured workflows, correctness
- Structural flaws
  - Taxonomy
  - Analyzing structural flaws
  - Introducing loops
- Literature
- Summary

Motivation

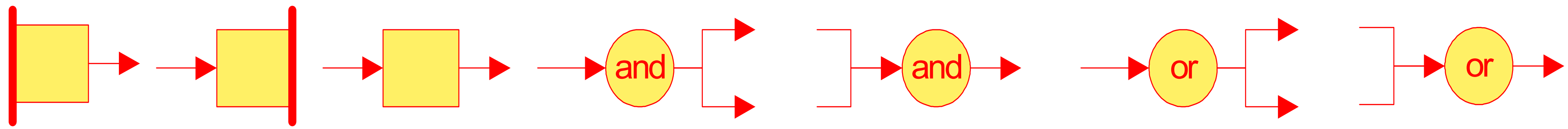
- Workflow technology: an important tool for businesses to integrate and automate business processes
- Process models
  - Precisely capture business requirements
  - Ensure successful workflow execution
  - A correct process model is without structural flaws
- How to verify process models?
  - Structuredness is one criterion
- Structured vs. unstructured workflows
  - Structured: correct, but restrictive
  - Unstructured: more expressive, but maybe incorrect – hard to verify

Objectives

- Analyze unstructured workflows using a general framework
  - Verify correctness
  - Detect structural flaws
    - Deadlocks
    - Multiple active instances of the same activity (or lack of synchronization)
  - Develop equivalent structured mappings (if possible)
    - Most workflow products impose structural constraints
    - Structured workflows are widely supported

Workflow Definition

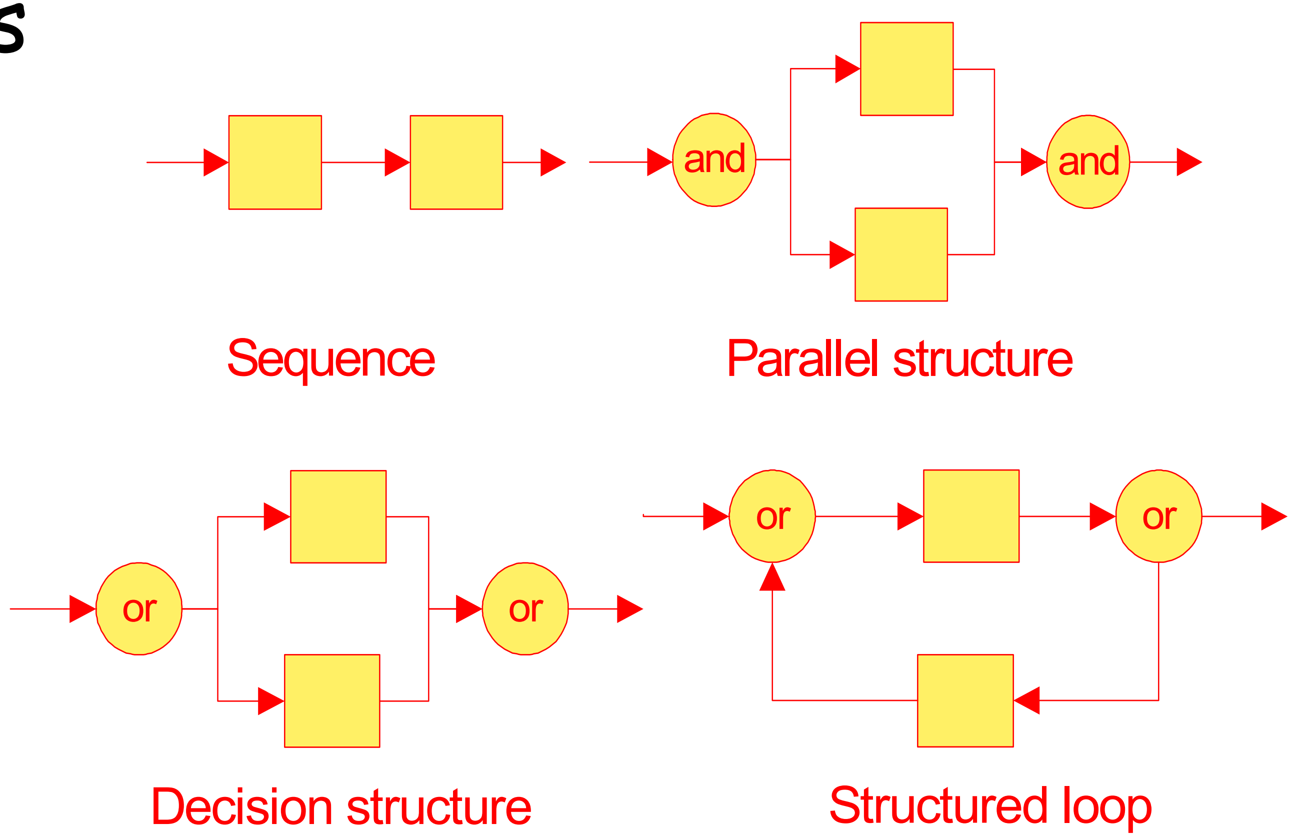
- A directed graph consisting of activities, arcs, and control elements
  - Control elements: **start**, **end**, **split-parallel**, **join-parallel**, **split-choice**, and **join-choice**
  - Connectivity: any node is in at least one path from the start node to the end node
- Semantics
  - Split-parallel, join-parallel
  - Split-choice: exclusive choice
  - Join-choice
    - Single execution
    - Multiple executions: may cause multiple instances



(a) Start (b) End (c) Activity (d) Split-parallel (e) Join-parallel (f) Split-choice (g) Join-choice

Structured Workflows

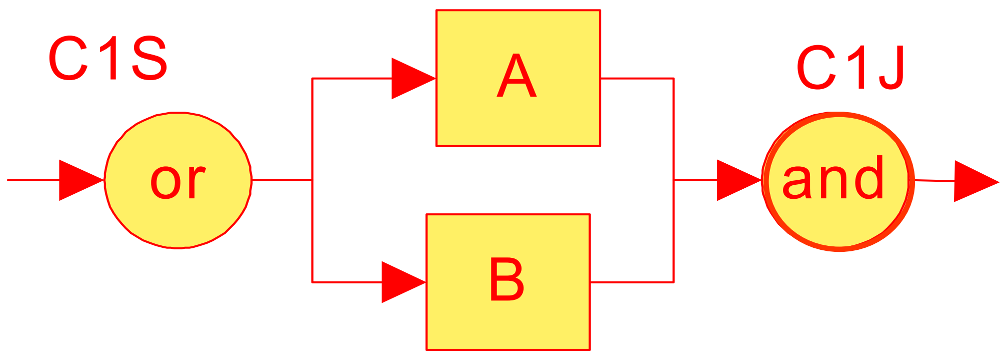
- Basic types
  - Sequence
  - Decision
  - Parallel
  - Loop



- They are correct and widely supported

Workflow Correctness

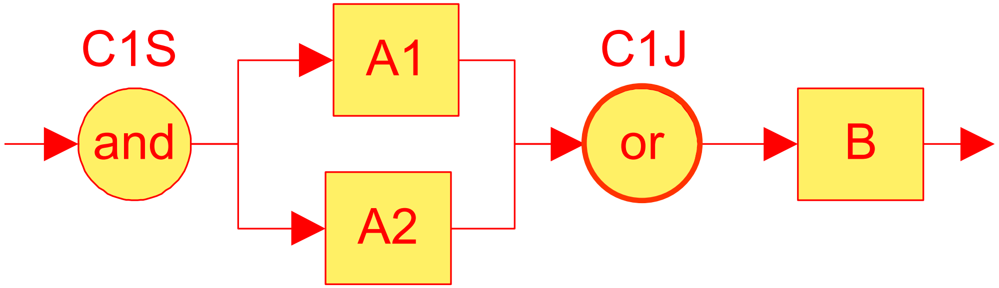
- Deadlocks: e.g. node C1J in wf1



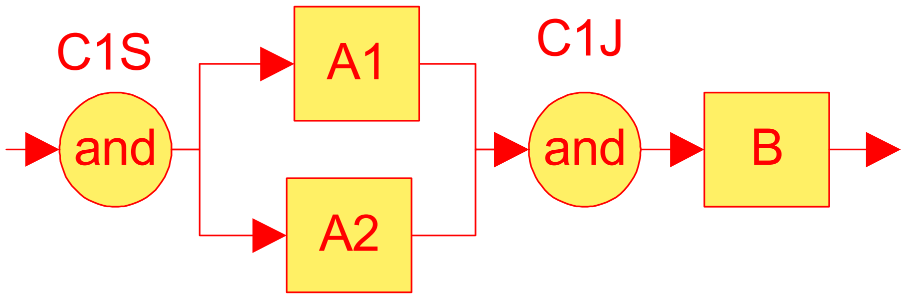
Workflow wf1

Workflow Correctness (contd.)

- Multiple active instances of the same activity
  - Semantics of **C1J** in wf2
    - Single execution
      - No structural flaws
      - Q-equivalent mapping: wf3
        - » wf2 can simulate wf3, but wf3 cannot simulate wf2
    - Multiple executions
      - multiple instances of **B** in wf2
- A well behaved workflow is without deadlock and multiple instances of the same activity



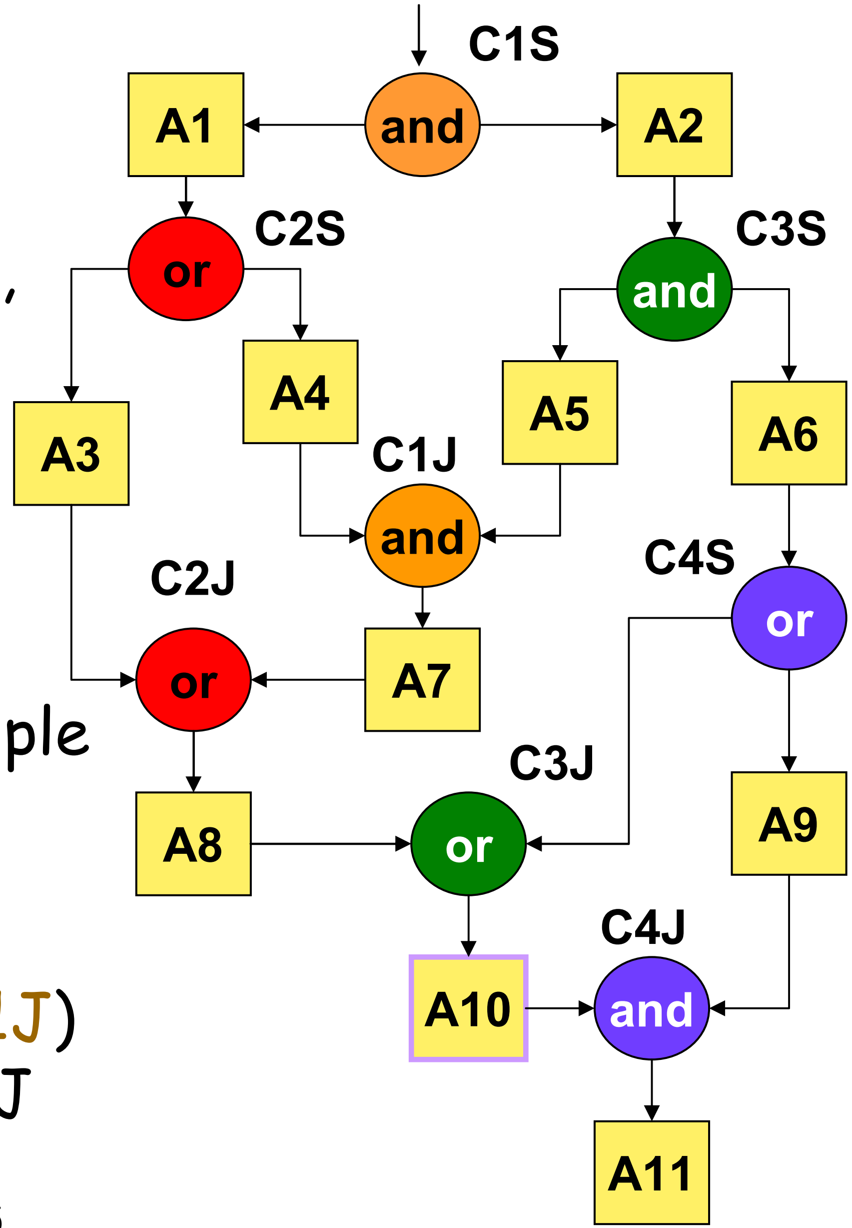
Workflow wf2



Workflow wf3

Taxonomy - Example

- Corresponding control elements
  - (C1S, C1J), (C2S, C2J), (C3S, C3J), (C4S, C4J)
- Mismatched pair
  - (or, and) -- (C4S, C4J), a deadlock at C4J
  - (and, or) -- (C3S, C3J), multiple instances of A10
- Improper nesting
  - 1st-order (C2S, C2J)[](C1S,C1J) may lead to a deadlock at C1J



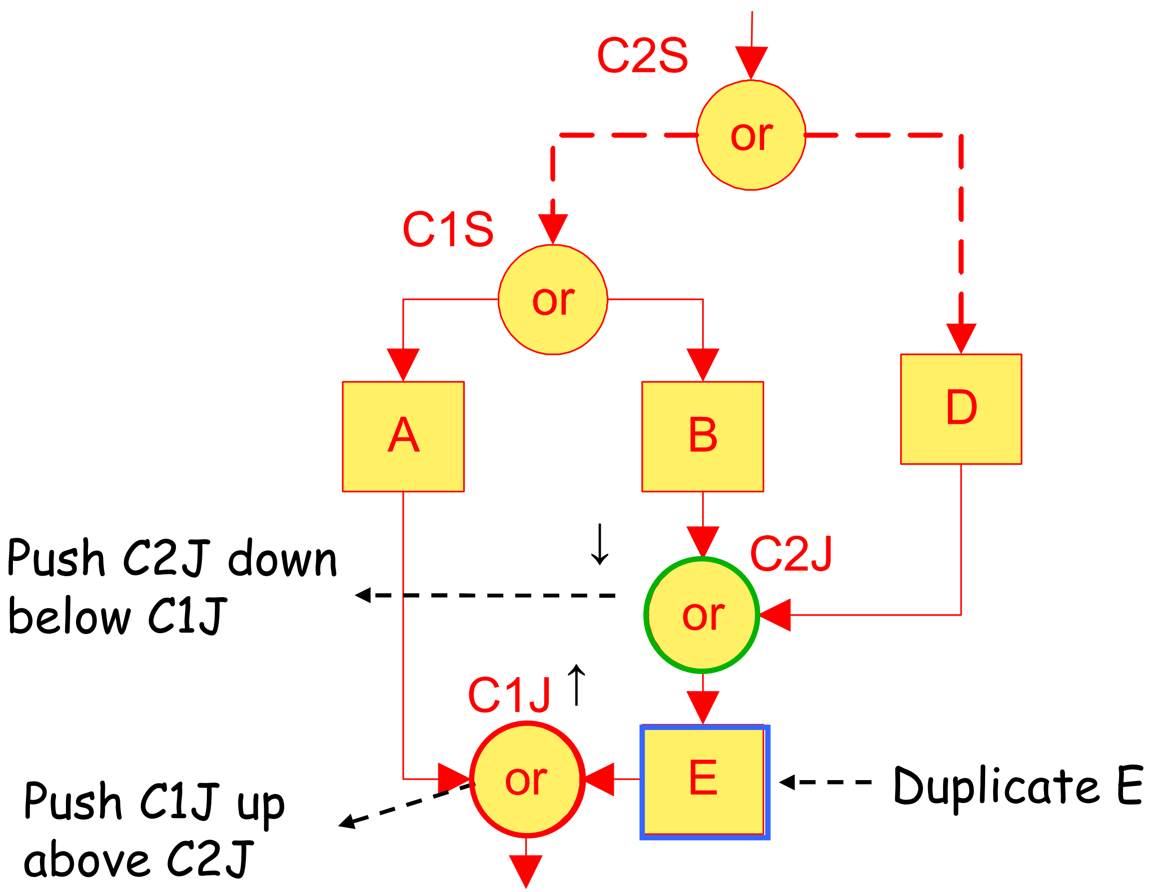
Taxonomy

- **Corresponding** control elements (s, j)
  - Two minimal paths starting from **s** first join at **j**
- **Mismatched pair**
  - (or, and): lead to deadlocks
  - (and, or): result in multiple instances
- **Improper nesting** $(u, v)^{[}{}_{]}(s, j)$
  - Pairs (u, v) (s, j) are matched; s (or j) is in a path from u to v, but j (or s) is not in this path
  - Nth-order improper nesting: there exist $(u, v)\ ^{[}{}_{]}\ (x_i, y_i)$, where $(x_i, y_i) \neq (s, j)$ and $i=1, 2, \ldots, n-1$
- A workflow is correct if all pairs of control elements are matched and properly nested!

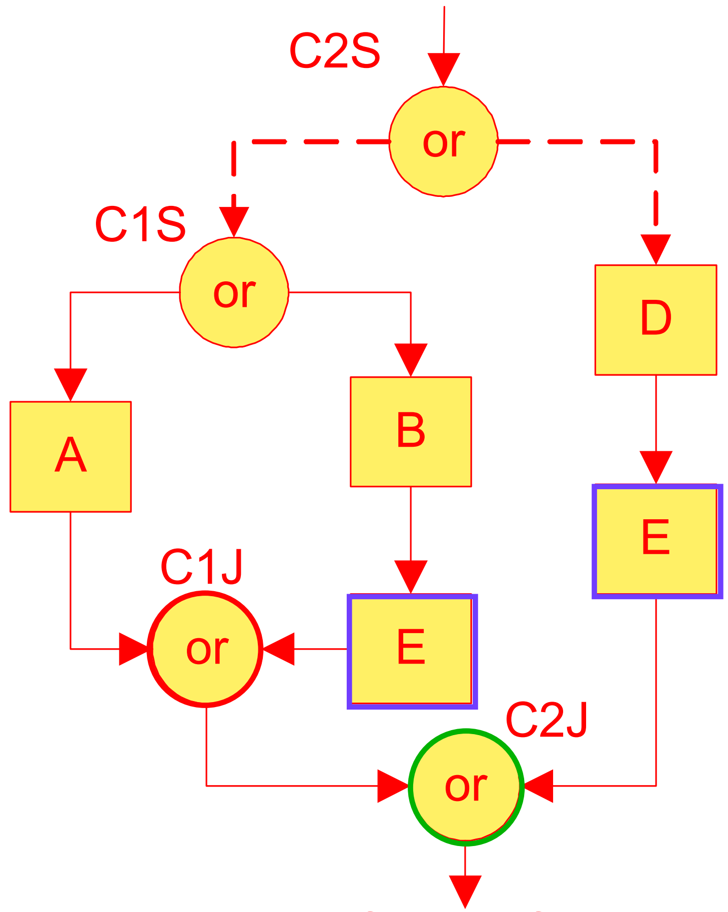
Analyzing Improper Nesting

- 1st order improper nesting
  - By case-by-case enumeration
- 2nd – and higher-order improper nesting
  - Reduce higher-order to 1st-order
  - Approach: switch adjacent join nodes and duplicate activities between them
  - Issue: switching may not always be possible (case-by-case analysis)

Analyzing Improper Nesting - Example



Workflow wf1 (Type 1)



Workflow wf2 (Mapping of wf1)

1st order Improper nesting

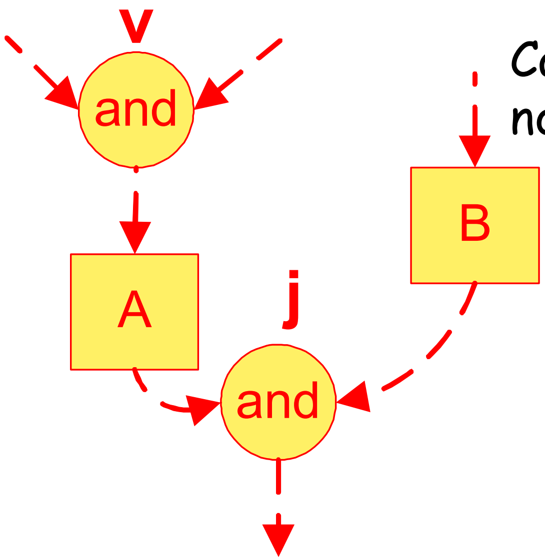
Type	(C1S	C1J)	(C2S	C2J)	Correctness issues	Structured transformation
1	OR	OR	OR	OR	well-behaved	Yes
2	OR	OR	OR	AND	deadlock	No
3	OR	OR	AND	OR	multiple instances	No
4	OR	OR	AND	AND	deadlock	No
5	AND	AND	OR	OR	deadlock	No
6	AND	AND	OR	AND	deadlock	No
7	AND	AND	AND	OR	multiple instances	q-equivalent mapping
7	AND	AND	AND	AND	well-behaved	No
9	OR	AND	OR	OR	deadlock	No
10	OR	AND	OR	AND	deadlock	No
11	OR	AND	AND	OR	deadlock	No
12	OR	AND	AND	AND	deadlock	No
13	AND	OR	OR	OR	multiple instances	No
14	AND	OR	OR	AND	deadlock	No
15	AND	OR	AND	OR	multiple instances	q-equivalent mapping
16	AND	OR	AND	AND	multiple instances	q-equivalent mapping

2nd- and higher-order improper nesting

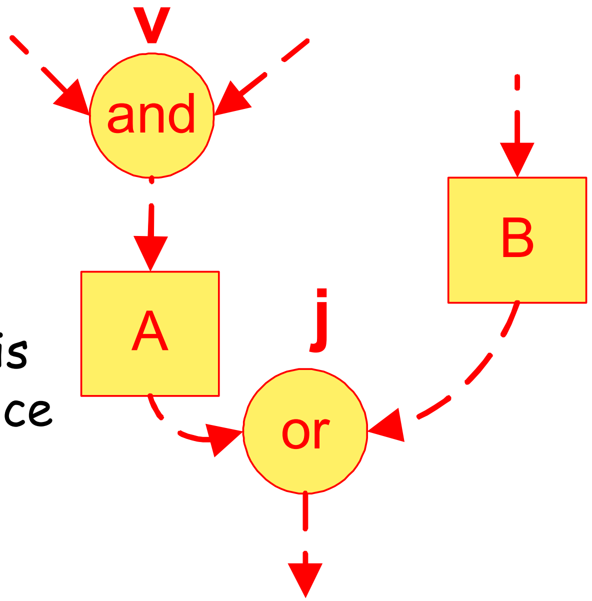
- Case by case analysis of adjacent join nodes
  - Case 1: two join-parallel nodes are adjacent
  - Case 2: a join-parallel node is adjacent to a join-choice node downstream
  - Case 3: two join-choice nodes are adjacent
  - Case 4: a join-parallel node is adjacent to a join-choice node upstream

  (Special case: overlapping structure)

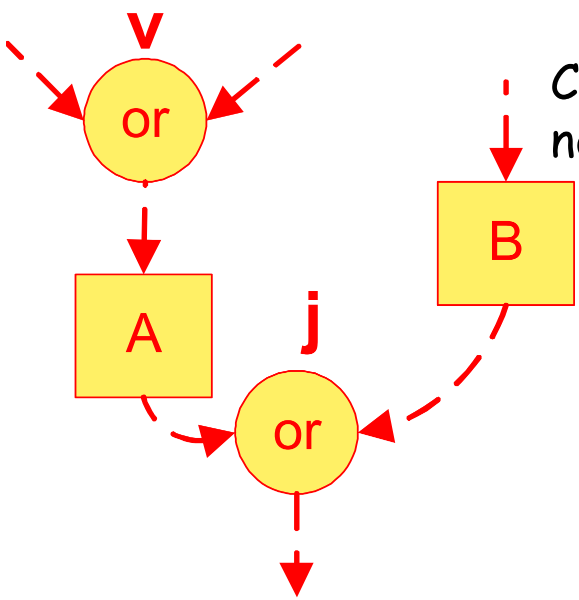
2nd- and higher-order improper nesting (contd.)



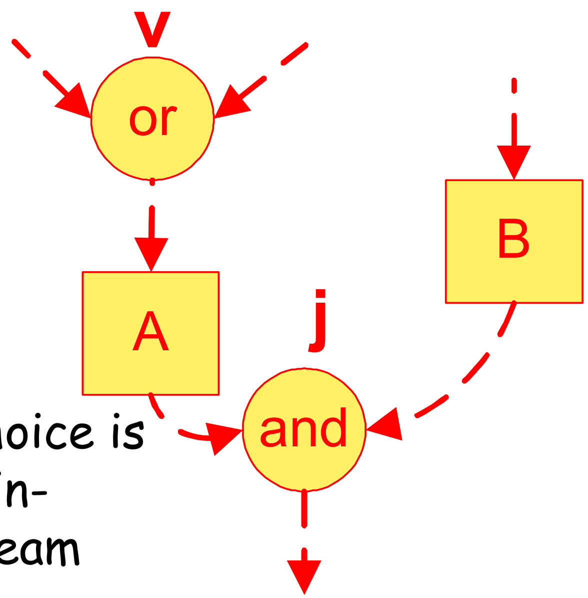
Case 1: two join-parallel nodes are adjacent



Case 2: a join-parallel is adjacent to a join-choice downstream

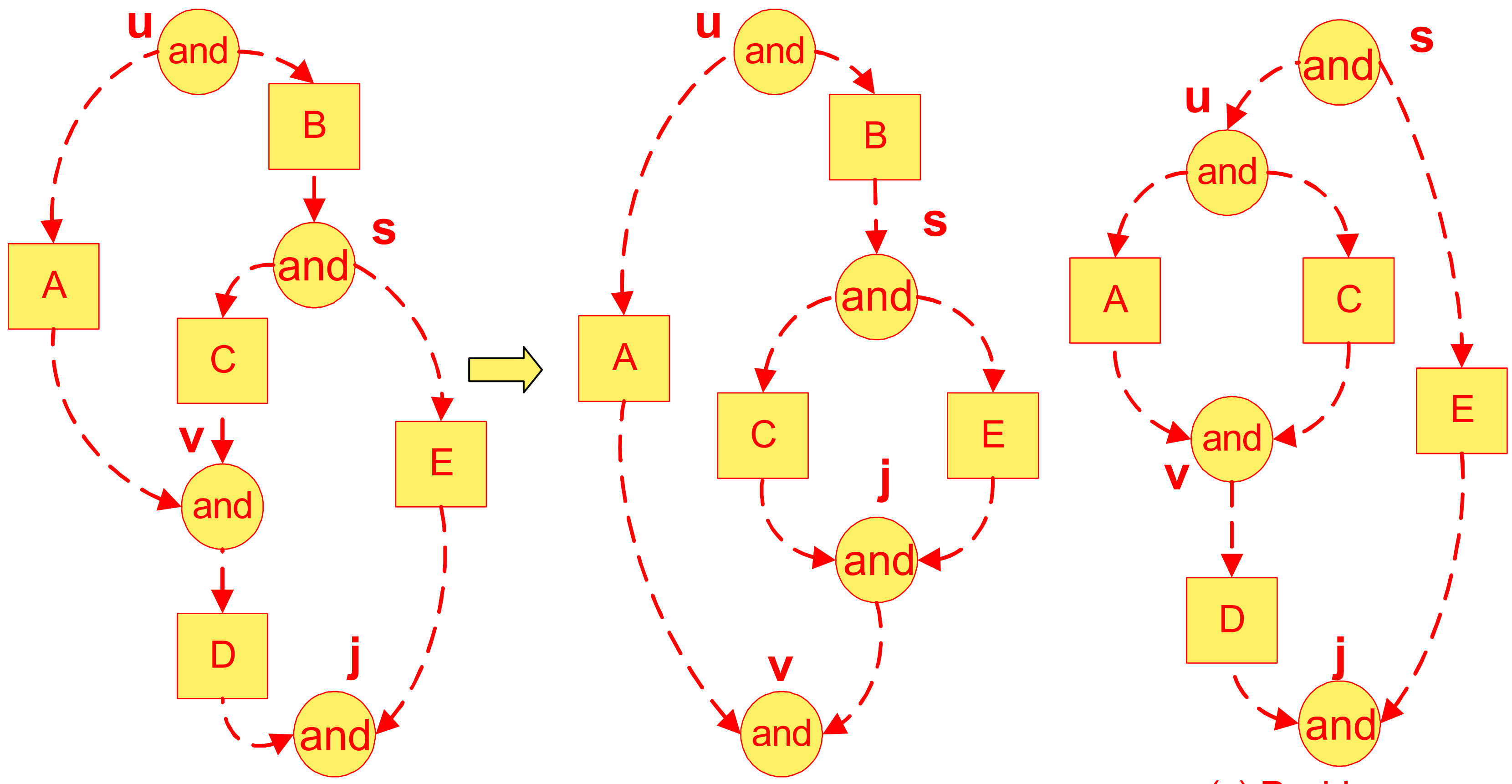


Case 3: two join-choice nodes are adjacent



Case 4: a join-choice is adjacent to a join-parallel downstream

Case 1



(a) Improper nesting example

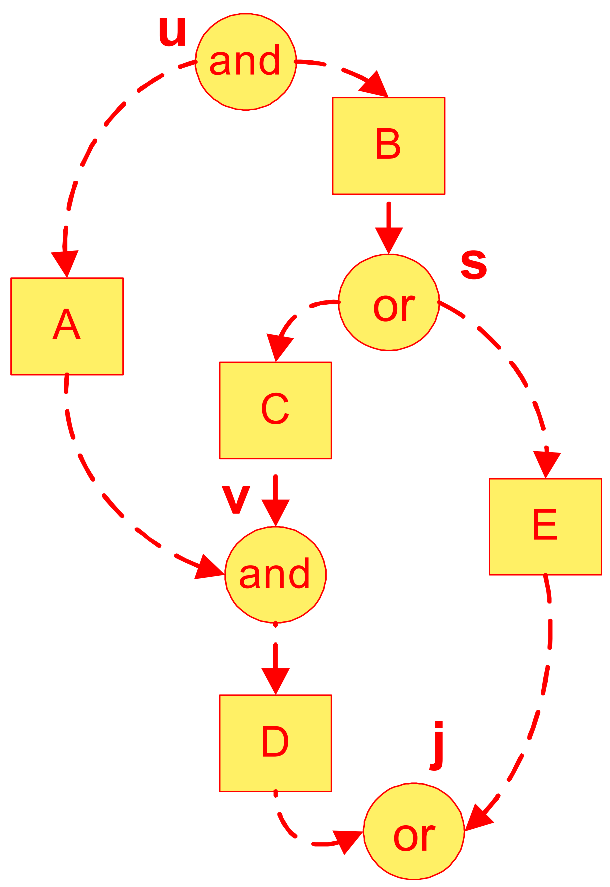
(b) Pushing ***v*** down (Node D removed)

(c) Pushing ***s*** up (Node B removed)

- No structured mapping if activities **B** and **D** present
  - To reduce higher order, switch **u** and **s** (or switch **v** and **j**)
  - But such switching is impossible if **B** and **D** are present
- A workflow with all AND nodes is well behaved

Case 2

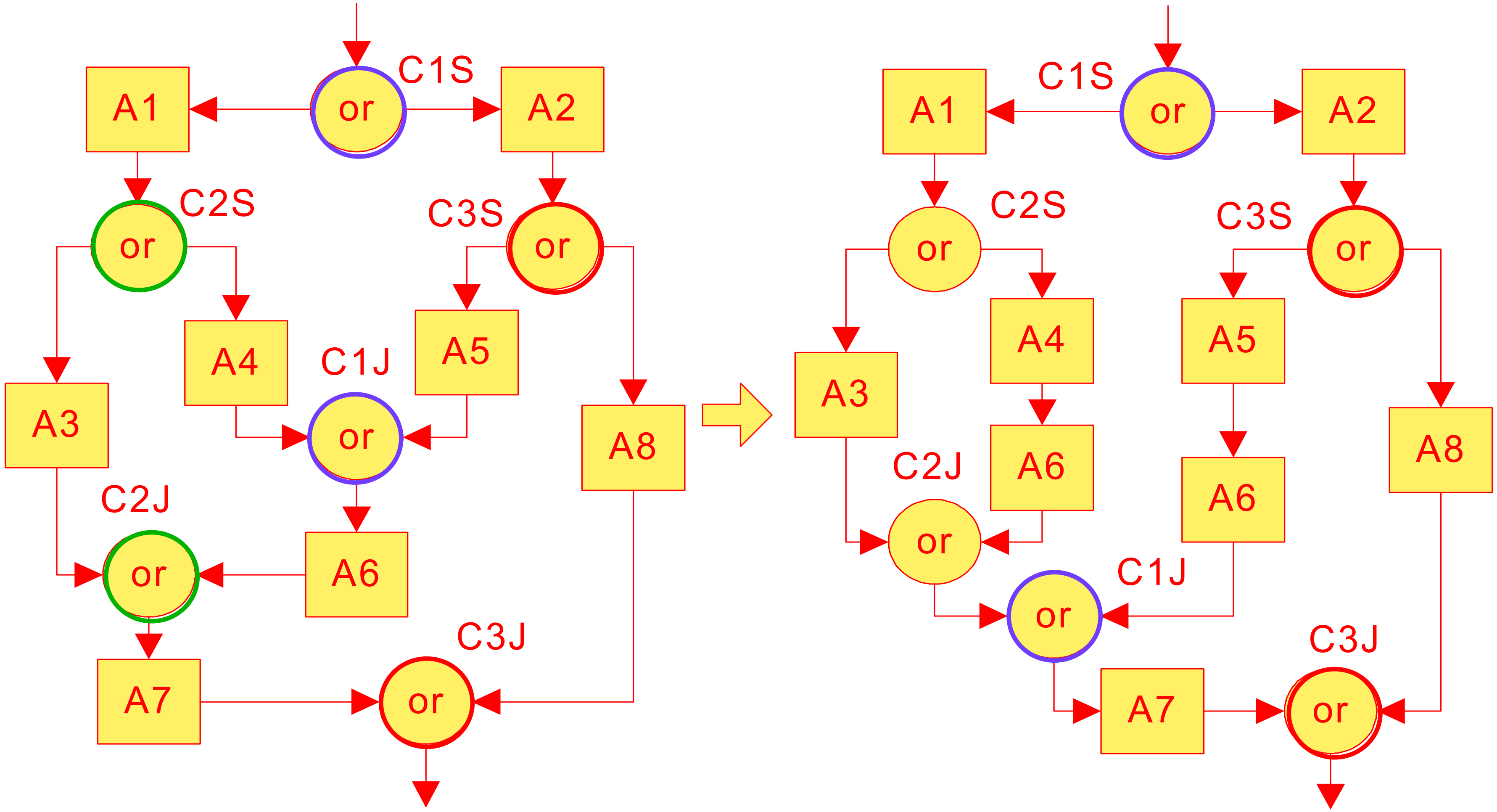
- Join-parallel is adjacent to a join-choice downstream
- Structured mapping does <u>not</u> exist
- Results in structural flaws – deadlock at v



Case 3

- Two join-choice nodes are adjacent
- If a workflow contains only OR nodes, it is well behaved and has a structured mapping
- Approach
  - Switch adjacent join-choice nodes
  - Duplicate activities between them
  - Reduce the order of improper nesting eventually

Case 3 - Example

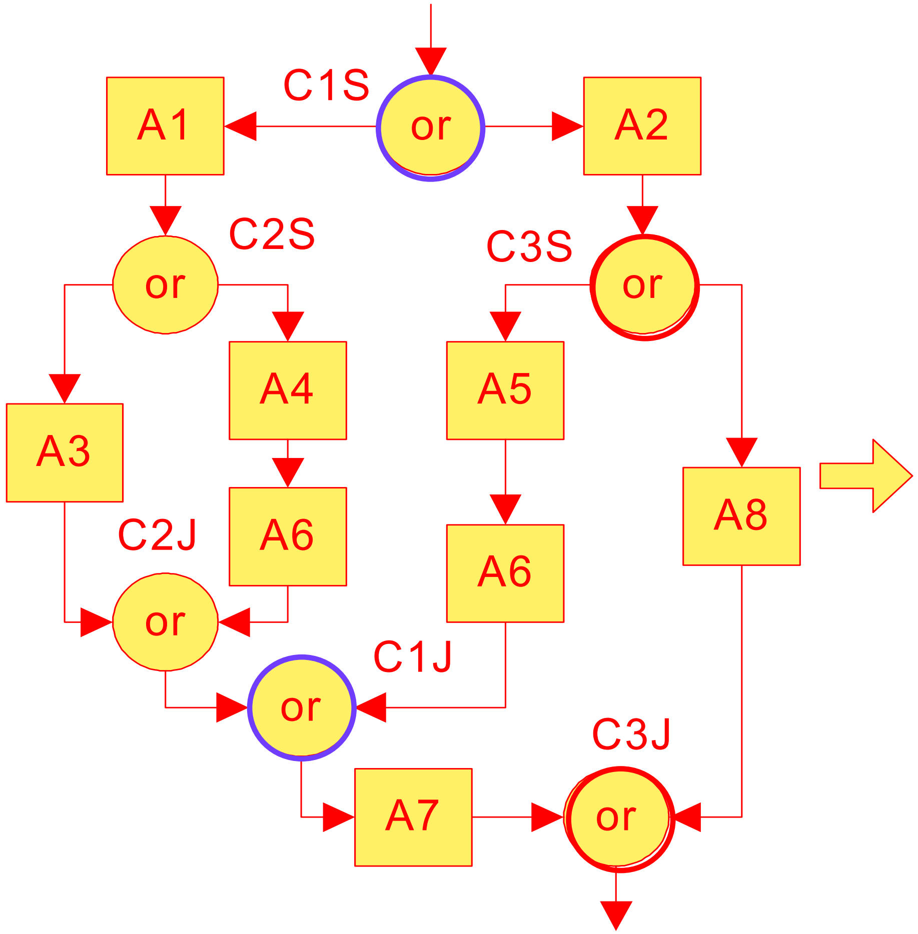


(a) Workflow wf1

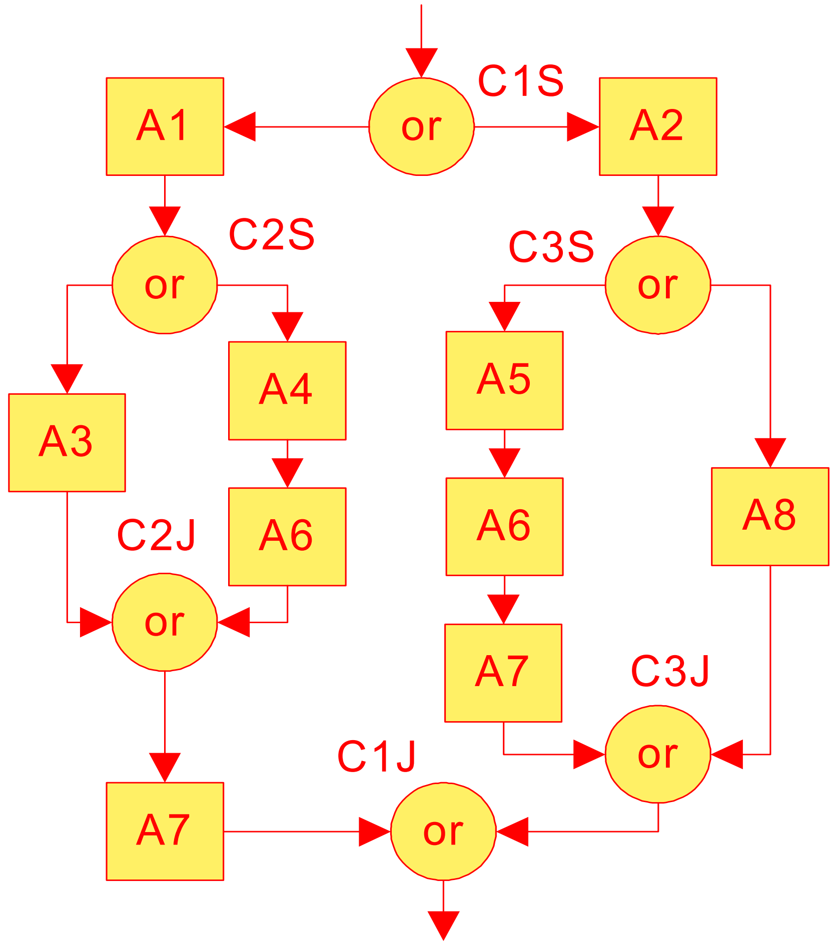
(b) Workflow wf2

Step 1: switch *C1J* and *C2J*, and duplicate *A6*

Case 3 - Example (Contd.)



(b) Workflow wf2



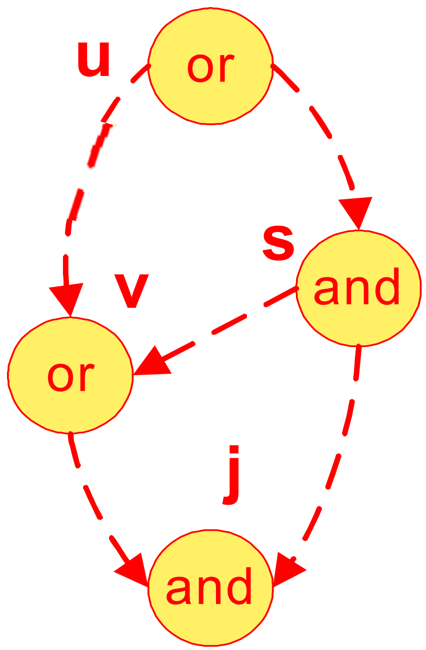
(c) Workflow wf3

Step 2: switch *C1J* and *C3J*, and duplicate *A7*

Structured mapping

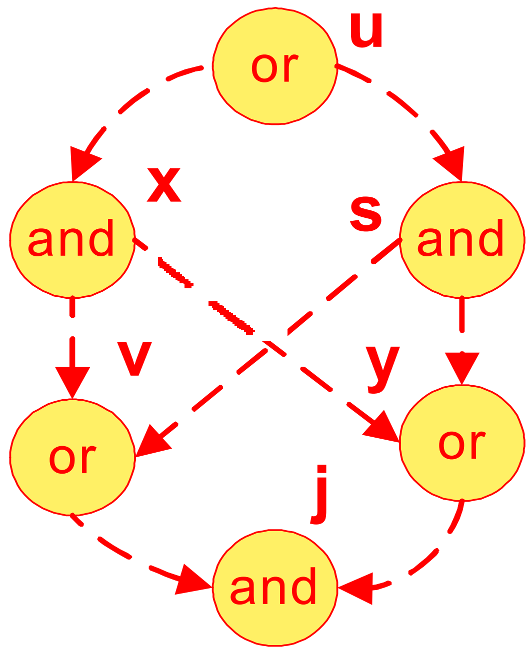
Case 4

- A join-parallel node is adjacent to a join-choice node upstream
- Typically it leads to deadlock at node j.
- <u>Special case</u>: an overlapping structure is correct and has a structured mapping



(a)

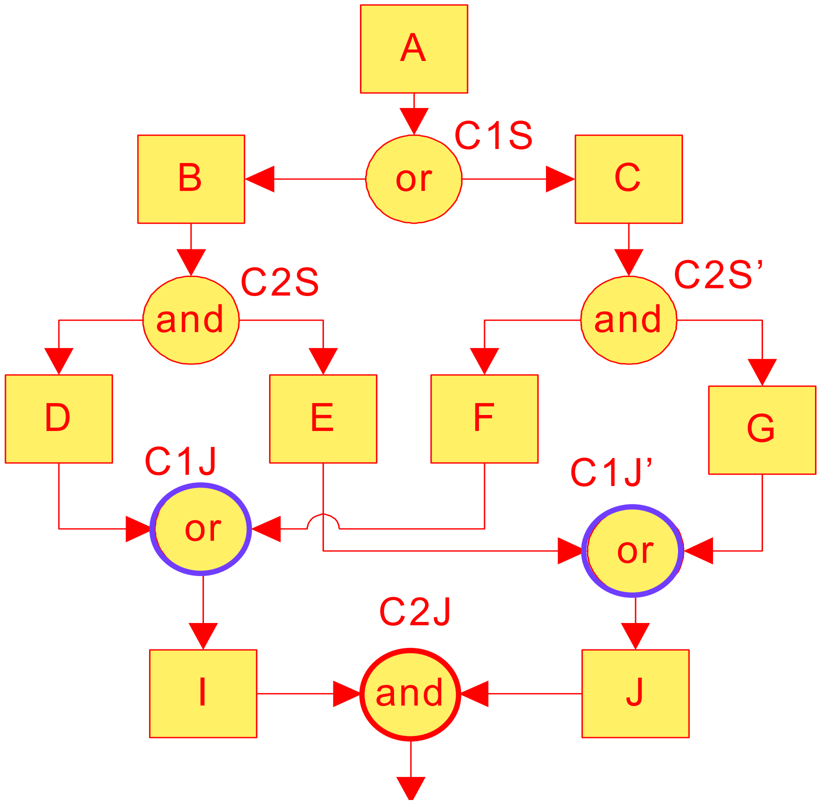
A join-parallel node is adjacent to only one join-choice node upstream

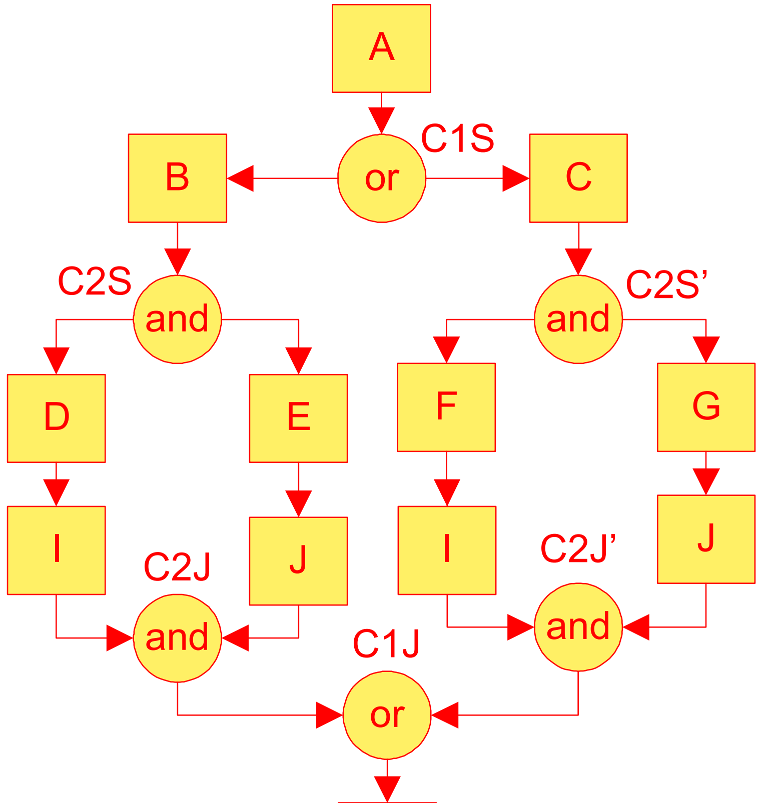


(b)

Overlapping structure: a join-parallel nodes is adjacent to <u>two</u> join-choice nodes upstream

Case 4 - Overlapping structure

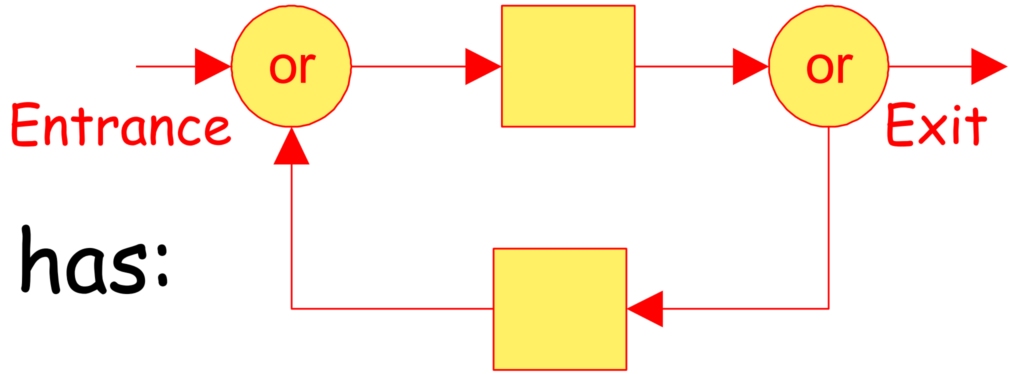




- Derive structured mapping: Push C2J up above C1J (C1J') as follows:
  1. Check paths from C2S (C2S') to C2J: DI, EJ, FI, GJ
  2. Find parallel paths (p and q): if p is taken, then q must be taken simultaneously, e.g., DI and EJ, FI and GJ
  3. Construct parallel structures using parallel path pairs
  4. Insert parallel structures inside OR pairs

Introducing Loops

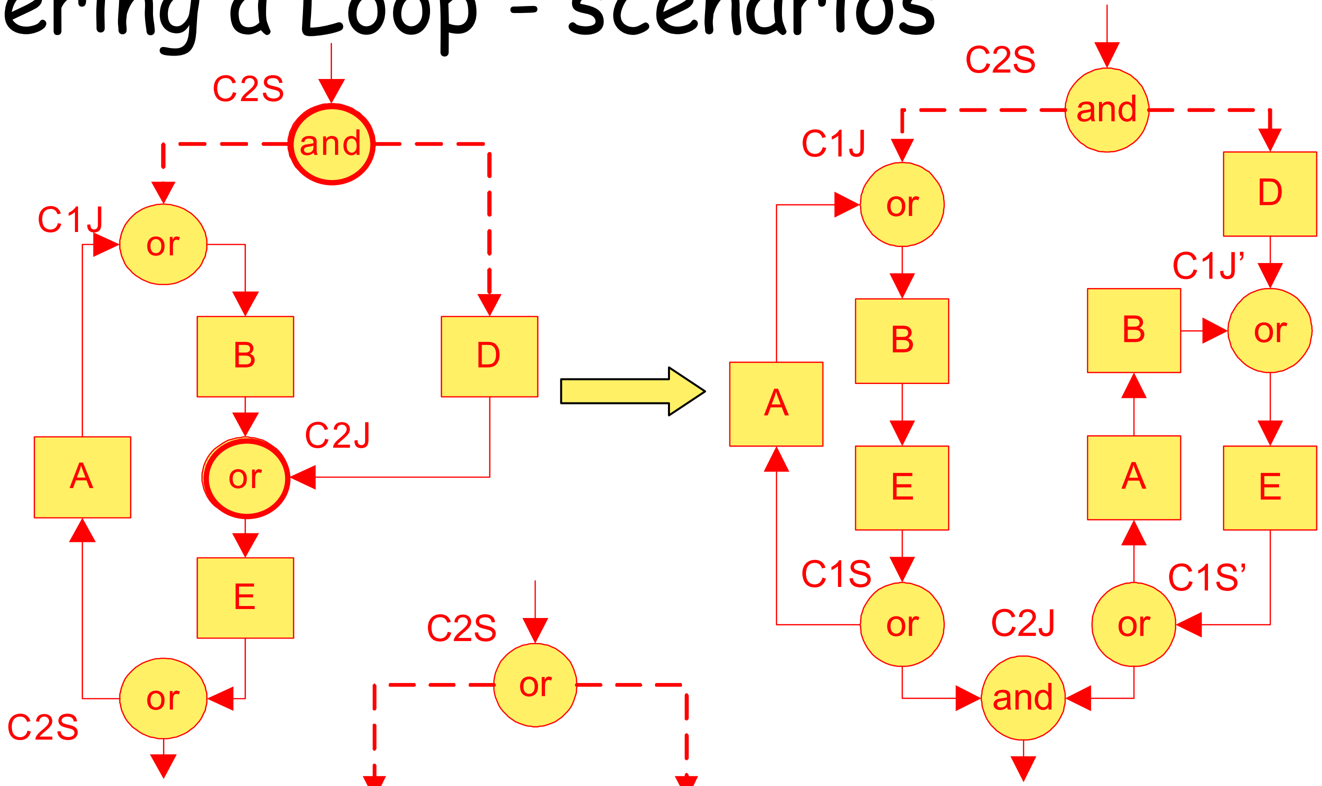
- So far, we only considered acyclic workflows
- A structured loop has:
  - one entrance
  - one exit
- An unstructured loop has:
  - additional entrances
  - additional exits



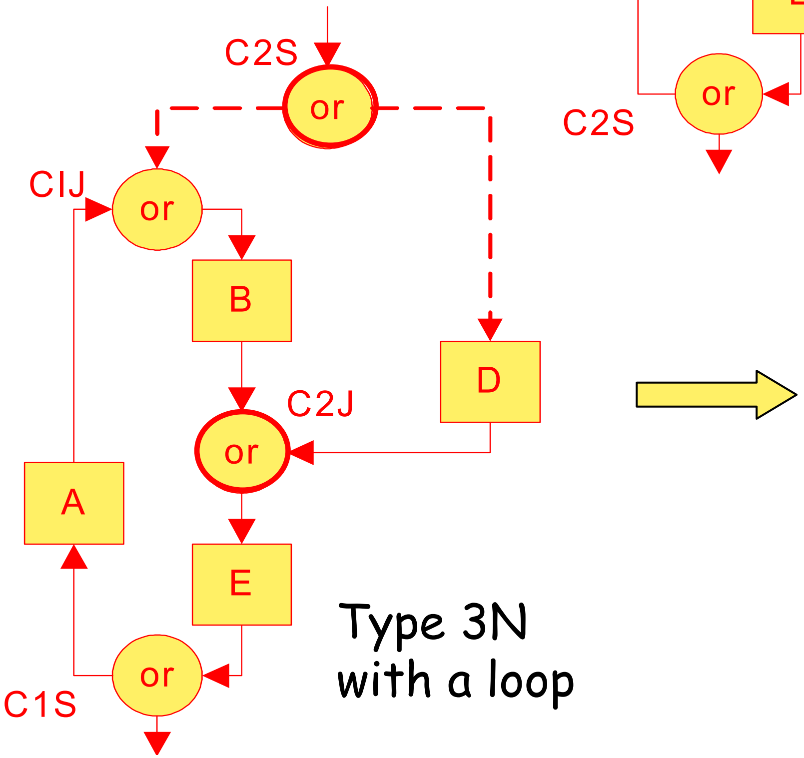
Structured loop

Entering a Loop - scenarios

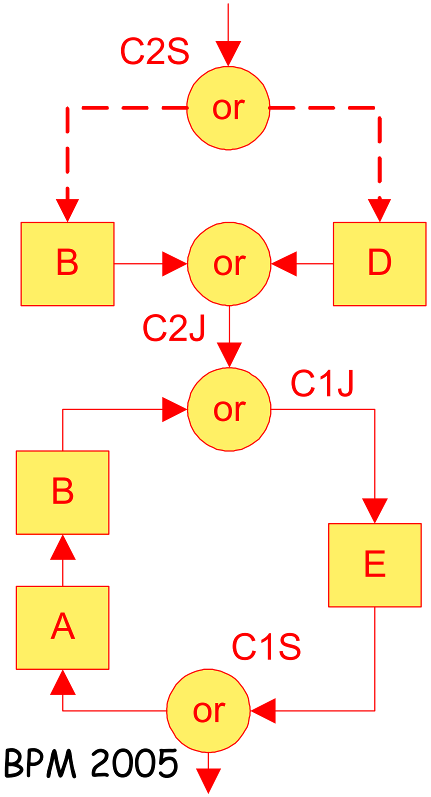
Type 1N with a loop



Q-equivalent mapping of Type 1N



Type 3N with a loop

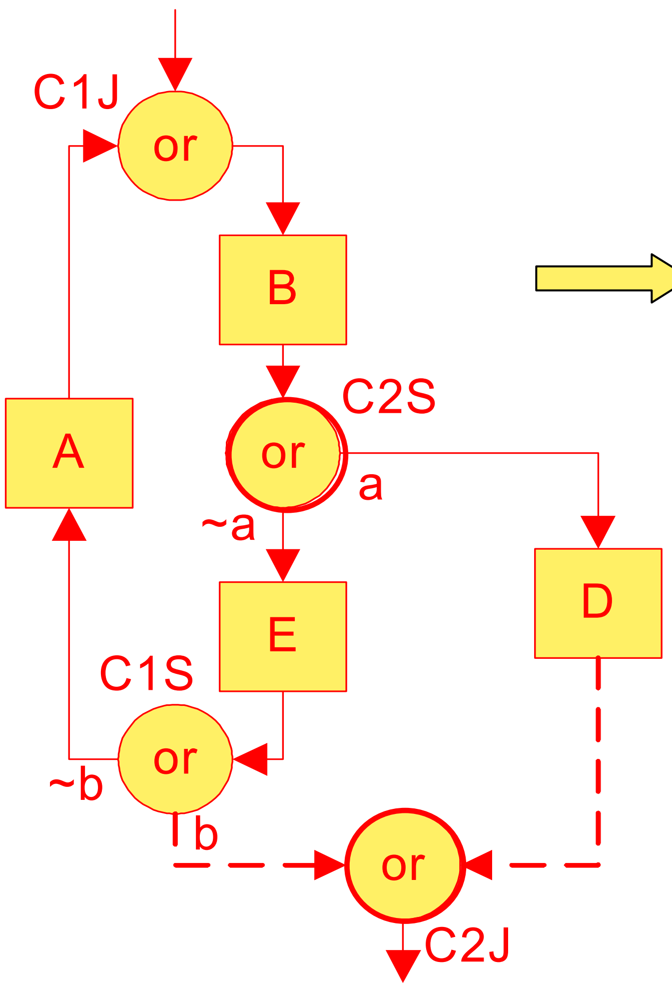


Equivalent mapping of Type 3N

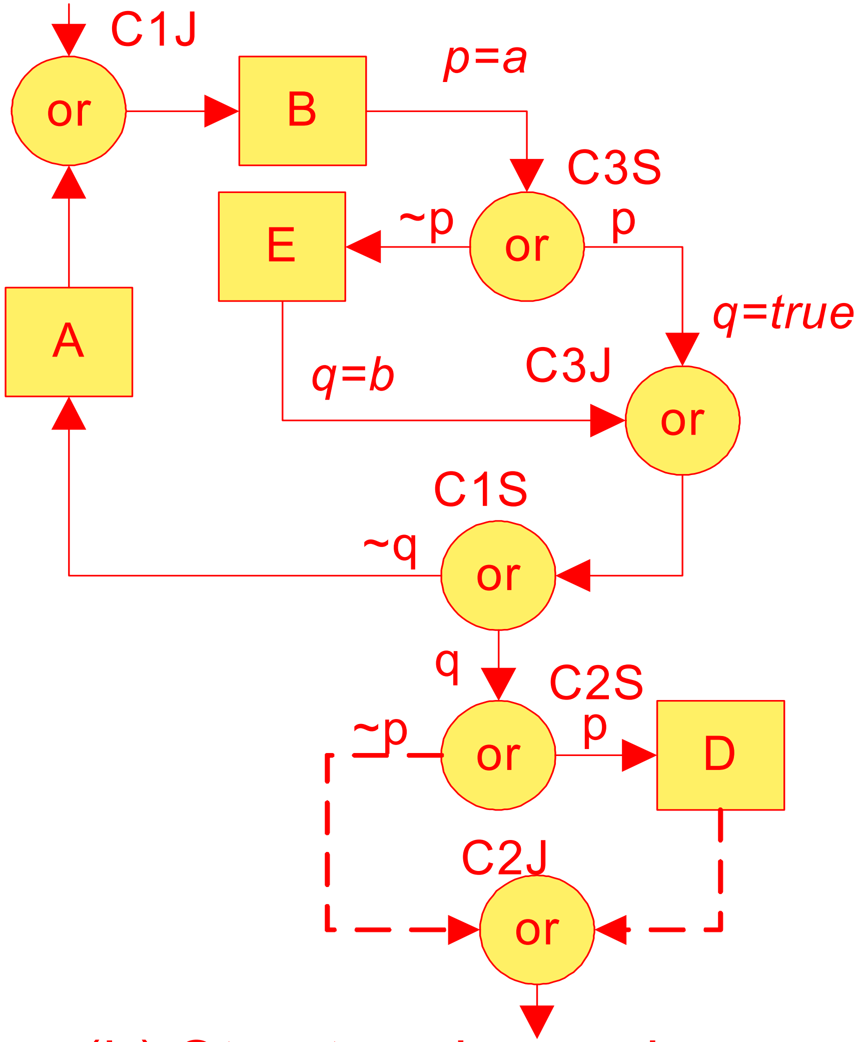
Entering a Loop

Type	(C2S	C2J)	Correctness issues	Structured Transformation
1N	AND	OR	multiple instances	q-equivalent mapping
2N	OR	AND	deadlock	No
3N	OR	OR	well-behaved	Yes
4N	AND	AND	deadlock	No

Exiting a loop - Example



(a) Type 3X (loop)



(b) Structured mapping

E<u>x</u>iting a loop

Type	**(C2S**	**C2J)**	**Correctness issues**	**Structured Transformation**
1X	AND	OR	multiple instances	No
2X	OR	AND	deadlock	No
3X	OR	OR	well-behaved	Yes (*auxiliary variable must be used*!)
4X	AND	AND	well-behaved	No

Literature Review

- Kiepuszewski et al, 2000, equivalence preserving transformation from unstructured workflows to structured ones
  - Based on examples
- Sadiq and Orlowska, 2000, graph reduction techniques to verify processes
  - Provide no clue for the causes of structural flaws
- Aalst et al, 1998, Petri net based approach
  - Natural structure of a workflow is lost
- Other approaches

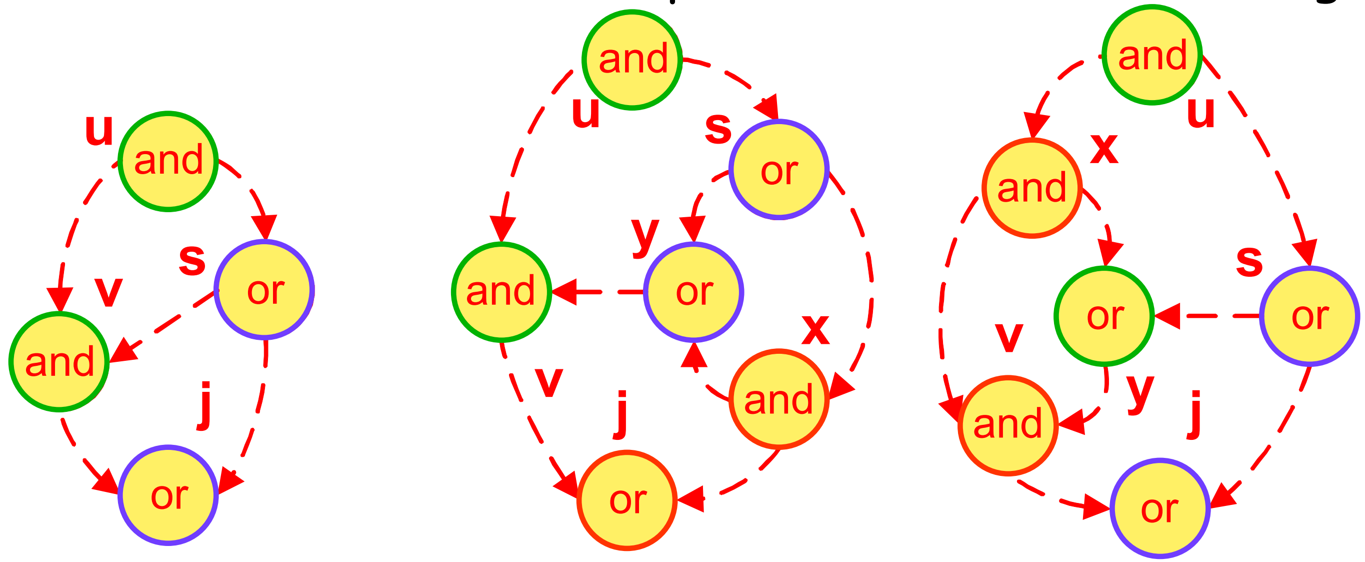
Conclusions and future work

- Formal taxonomy of unstructured workflow patterns
  - Mismatched pairs
    - (and, or) : multiple instances, q-equivalent mapping
    - (or, and) : deadlocks
  - Improper nesting
    - Some are well behaved: e.g., a workflow with all AND nodes
    - Some have structured mappings: e.g., a workflow with all OR nodes, overlapping structure
      - Structured mappings are an implementation workaround for tools that do not support unstructured workflows
    - Some lead to deadlocks
- Currently working on:
    - More general results on correctness
    - Algorithm to check errors in workflow and correct them

Correctness of Cases 1 & 2

- Case 1: If a workflow with only AND nodes is *well behaved*
- Case 2: An improper nesting $(u, v)^{[}{}_{]}(s, j)$, where (u, v) is an AND pair, (s, j) is an OR pair, and s is adjacent to u upstream, leads to a deadlock at node v.
  - Proof: When the right outgoing arc of s is selected, in order not to lead to a deadlock at v, a flow must be divided
    - either from the right outgoing path of s to v -- Figure (b),
    - or from the left outgoing path of u to v -- Figure (c)

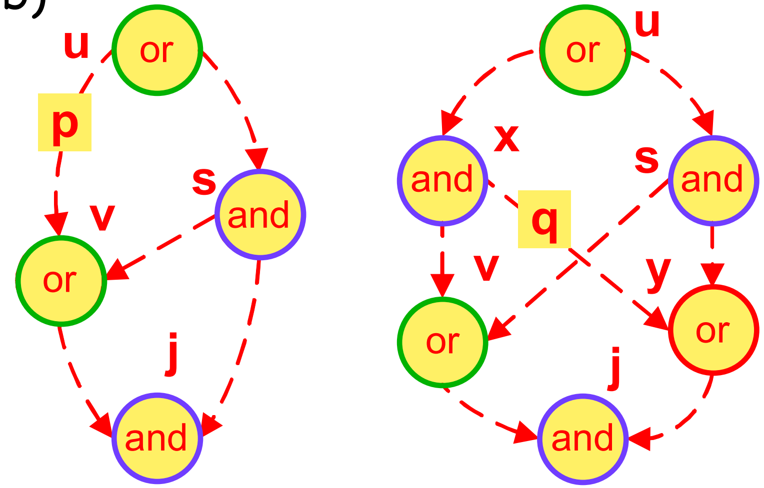
  In both situations, the correspondences have been changed.



(a) (b) (c)

Correctness of Cases 3 & 4

- Case 3: A workflow with all OR nodes is *well behaved* and has a structured mapping.
- Case 4 (more general): An improper nesting (*u*, *v*)$^{[}{}_{]}$(*s*, *j*), where (*u*, *v*) is an OR pair, (*s*, *j*) is an AND pair, and *j* is adjacent to *v* upstream, *leads to a deadlock at node j*, unless this improper nesting is a part of an overlapping structure
  - Proof: *p* cannot pass node *s*; otherwise, *u* will correspond to some other join node (including *s*) instead of *v*. So, if *p* is selected, no flow will come out of *s*. Therefore, in order not have a deadlock at *j*, a flow must be divided from path *p* to *j*, i.e., path *q* exists – an overlapping structure shown in Figure (b)

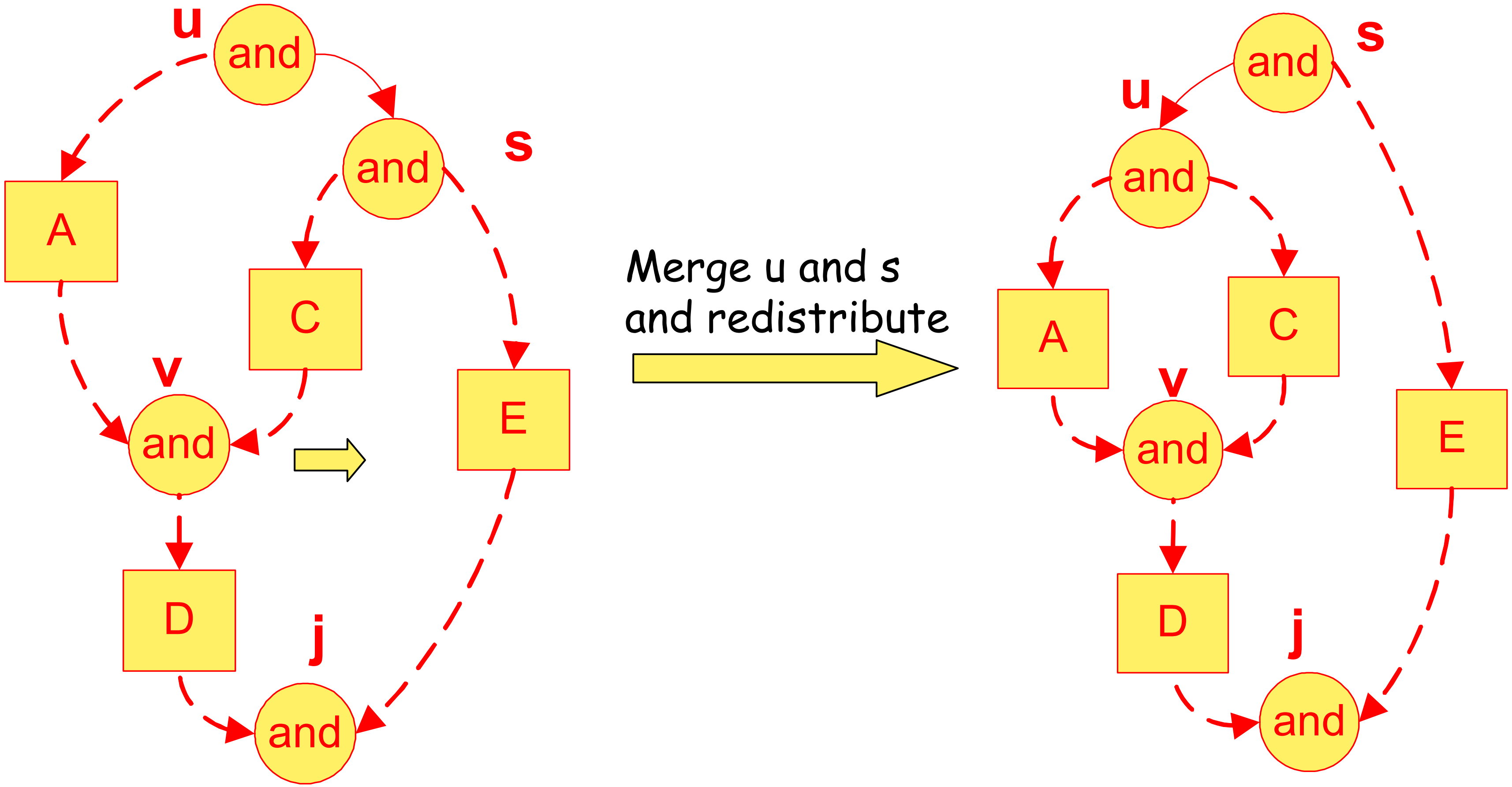


(a)

(b)

Case 1 - Special case (B not present)

- If B (or D) not present, structured mappings exists



(a) Improper nesting example

(b) Structured mapping