# Improved Layout for Data Flow Diagrams with Port Constraints

Lars Kristian Klauske[2] Miro Spönemann[1]
*Christoph Daniel Schulze*[1] Reinhard von Hanxleden[1]

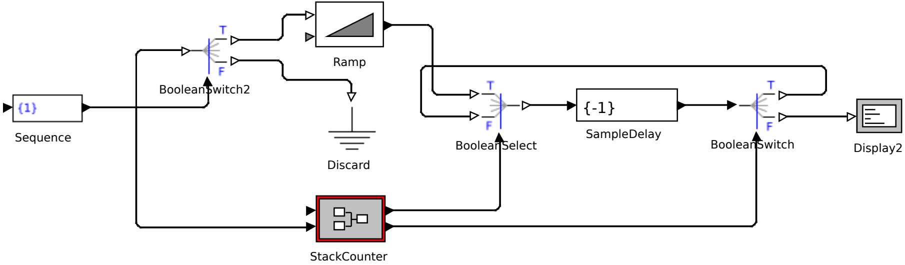
[1] Real-Time Systems and Embedded Systems Group
Department of Computer Science
Christian-Albrechts-Universität zu Kiel

[2] Daimler Center for Automotive Information Technology Innovations, Berlin

July 3rd / Diagrams 2012

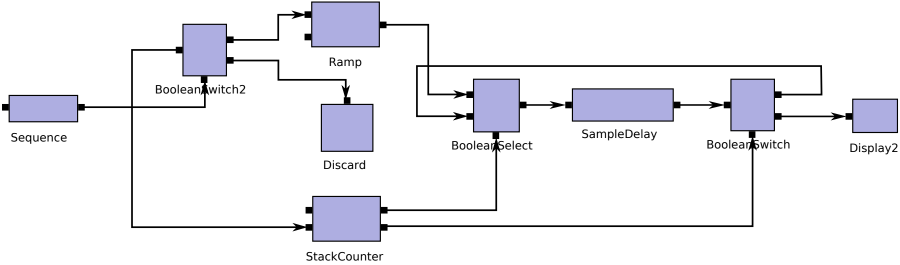
# Port-Based Data Flow Diagrams

## A Ptolemy Actor Diagram (Edward A. Lee, UC Berkeley)



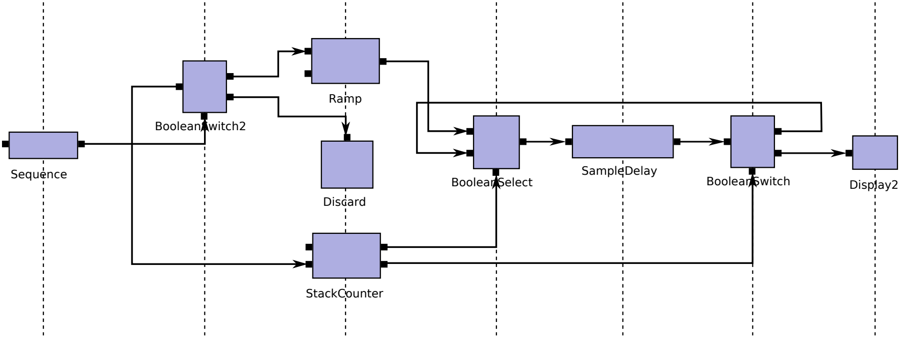
# Port-Based Data Flow Diagrams

## A Ptolemy Actor Diagram (Edward A. Lee, UC Berkeley)



# Port-Based Data Flow Diagrams

## A Ptolemy Actor Diagram (Edward A. Lee, UC Berkeley)



# Outline

1. The Importance of Port Support
   - Preliminaries
   - Inverted Ports
   - Side Ports
   - Putting It All Together
2. Demo Time!
3. Evaluation
4. Conclusion

# Outline

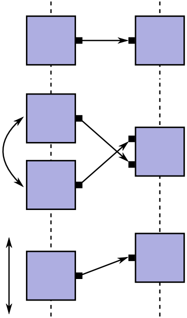
1. The Importance of Port Support
   - Preliminaries
   - Inverted Ports
   - Side Ports
   - Putting It All Together
2. Demo Time!
3. Evaluation
4. Conclusion

# Layer-Based Layout Algorithms

## Based on Sugiyama et al. (1981)

Five phases:

2. Layer assignment
3. Crossing reduction
4. Node placement

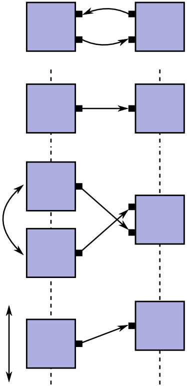


# Layer-Based Layout Algorithms

## Based on Sugiyama et al. (1981)

Five phases:

1. Cycle removal
2. Layer assignment
3. Crossing reduction
4. Node placement

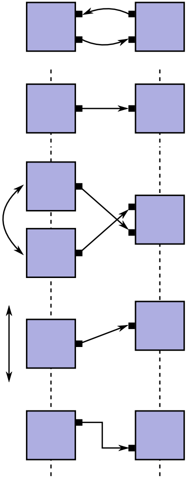


# Layer-Based Layout Algorithms

Based on Sugiyama et al. (1981)

Five phases:

1. Cycle removal
2. Layer assignment
3. Crossing reduction
4. Node placement
5. Edge routing

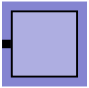


# Port Constraints

Based on Spönemann, Fuhrmann, von Hanxleden, Mutzel (2009)

Five levels of freedom:

1. Free

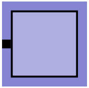


# Port Constraints

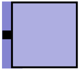
## Based on Spönemann, Fuhrmann, von Hanxleden, Mutzel (2009)

Five levels of freedom:

1. Free



2. Fixed Sides

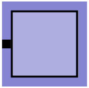


# Port Constraints

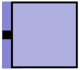
Based on Spönemann, Fuhrmann, von Hanxleden, Mutzel (2009)

Five levels of freedom:

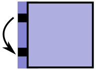
1. Free



2. Fixed Sides



3. Fixed Order

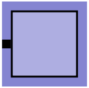


# Port Constraints

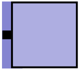
Based on Spönemann, Fuhrmann, von Hanxleden, Mutzel (2009)

Five levels of freedom:

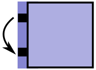
1. Free



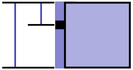
2. Fixed Sides



3. Fixed Order



4. Fixed Ratio

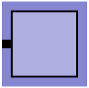


# Port Constraints

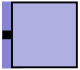
Based on Spönemann, Fuhrmann, von Hanxleden, Mutzel (2009)

Five levels of freedom:

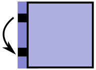
1. Free



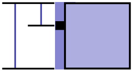
2. Fixed Sides



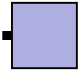
3. Fixed Order



4. Fixed Ratio



5. Fixed Positions



# Related Work on Port-Based Diagrams

## A Selection of Publications

- Gansner, Koutsofios, North, Vo (1993)
  - Ports considered during node placement, not crossing reduction
- Sander (1994)
  - Routing of side port edges encapsulated in single dummy node
- Waddle (2001)
  - Considers order of ports during crossing reduction
- Spönemann, Fuhrmann, von Hanxleden, Mutzel (2009)
  - Introduction of port constraints and a basic layout algorithm
- Klauske, Dziobek (2010, 2011)
  - Port positions depend on (modifiable) node size

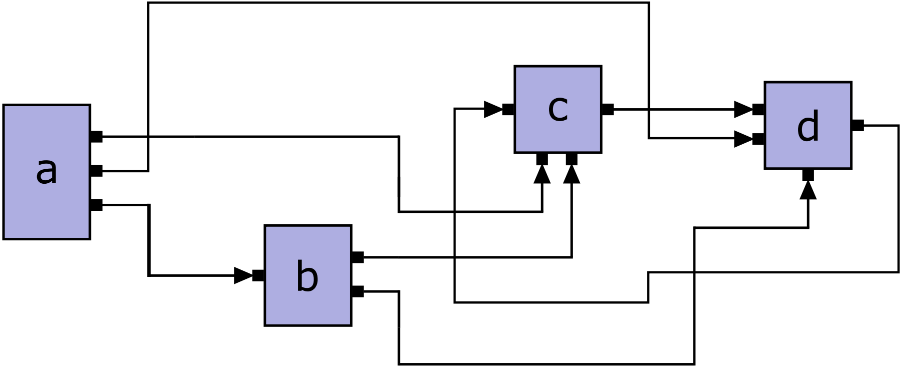
# Related Work on Port-Based Diagrams

## A Selection of Publications

- Gansner, Koutsofios, North, Vo (1993)
  - Ports considered during node placement, not crossing reduction
- Sander (1994)
  - Routing of side port edges encapsulated in single dummy node
- Waddle (2001)
  - Considers order of ports during crossing reduction
- Spönemann, Fuhrmann, von Hanxleden, Mutzel (2009)
  - Introduction of port constraints and a basic layout algorithm
- Klauske, Dziobek (2010, 2011)
  - Port positions depend on (modifiable) node size
- Wybrow, Marriott, Stuckey (2012)
  - More targeted towards network diagrams
  - However, application to data flow diagrams could be interesting

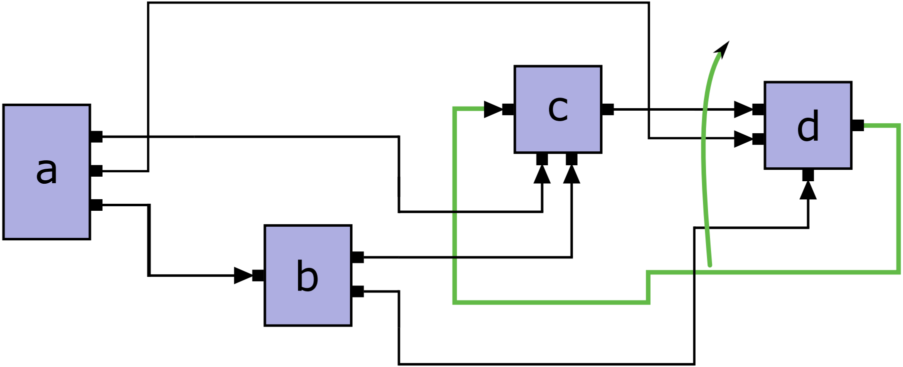
# Challenges With Port-Based Diagrams

## A Layout Produced by Our Previous Algorithm



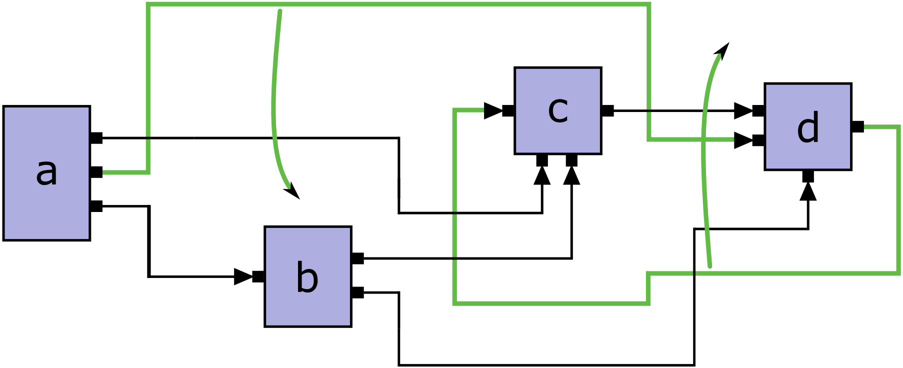
# Challenges With Port-Based Diagrams

## A Layout Produced by Our Previous Algorithm



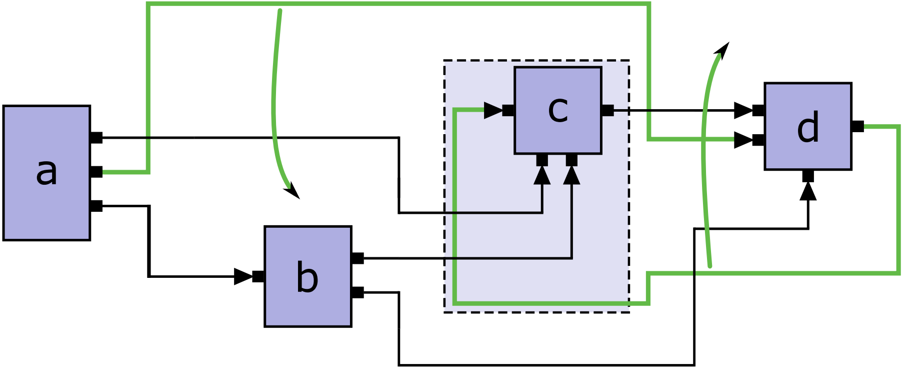
# Challenges With Port-Based Diagrams

## A Layout Produced by Our Previous Algorithm



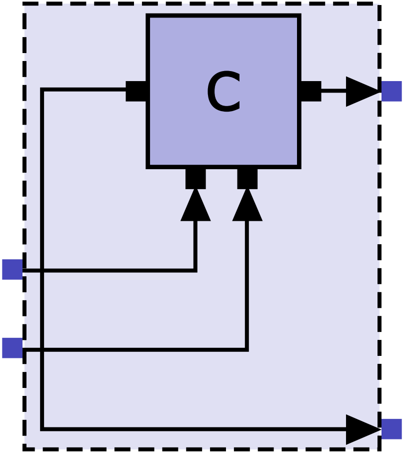
# Challenges With Port-Based Diagrams

## A Layout Produced by Our Previous Algorithm



# The Problem: Node-Local Edge Routing

## Making Strange Nodes Look Perfectly Ordinary

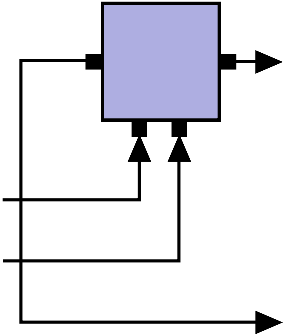


# Outline

1. The Importance of Port Support
   - Preliminaries
   - Inverted Ports
   - Side Ports
   - Putting It All Together
2. Demo Time!
3. Evaluation
4. Conclusion

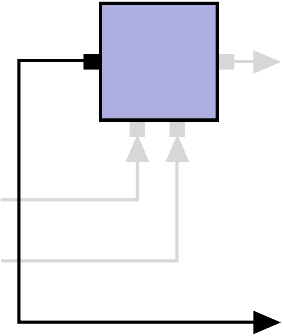
# Handling Inverted Ports

## An Approach Based on Dummy Nodes



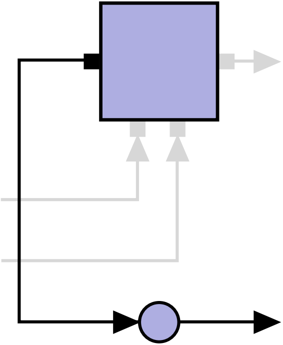
# Handling Inverted Ports

## An Approach Based on Dummy Nodes



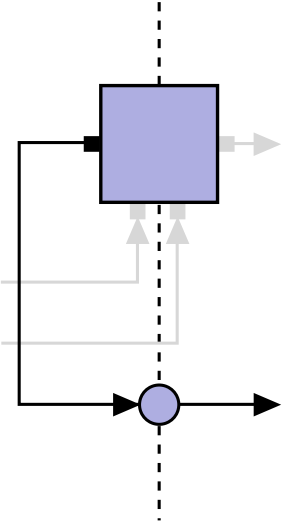
# Handling Inverted Ports

## An Approach Based on Dummy Nodes



# Handling Inverted Ports

## An Approach Based on Dummy Nodes

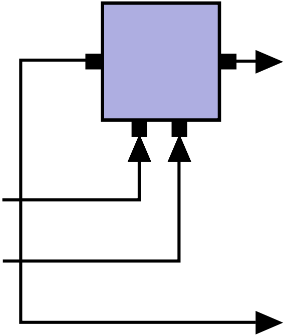


# Outline

1. The Importance of Port Support
   - Preliminaries
   - Inverted Ports
   - Side Ports
   - Putting It All Together
2. Demo Time!
3. Evaluation
4. Conclusion

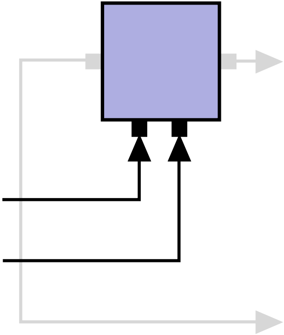
# Basic Handling of Side Ports

## Again Based on Dummy Nodes



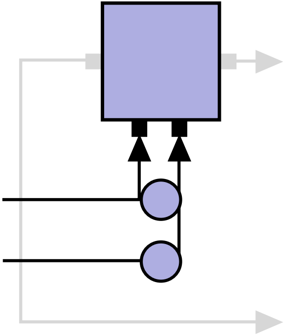
# Basic Handling of Side Ports

## Again Based on Dummy Nodes



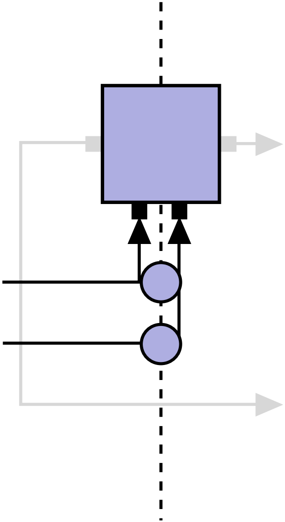
# Basic Handling of Side Ports

## Again Based on Dummy Nodes



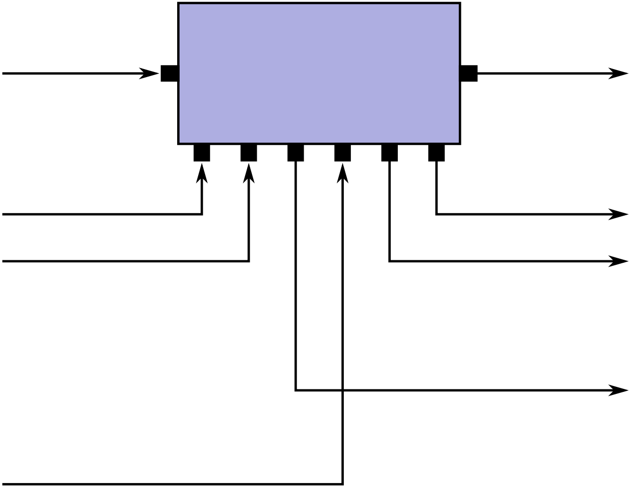
# Basic Handling of Side Ports

## Again Based on Dummy Nodes



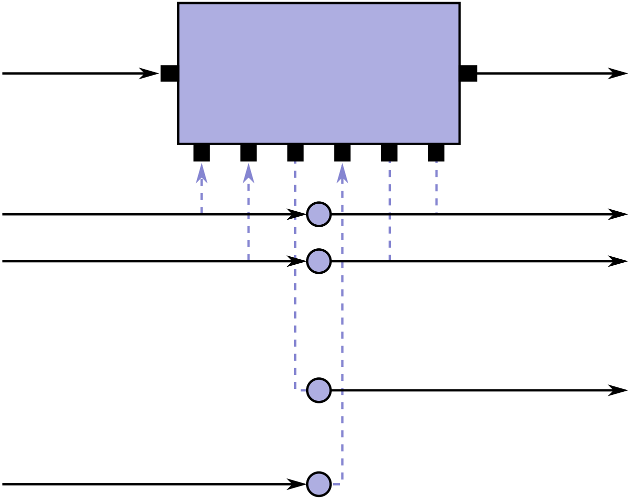
# Creating Dummy Nodes for Side Ports

## Local Optimization of Edge Crossings



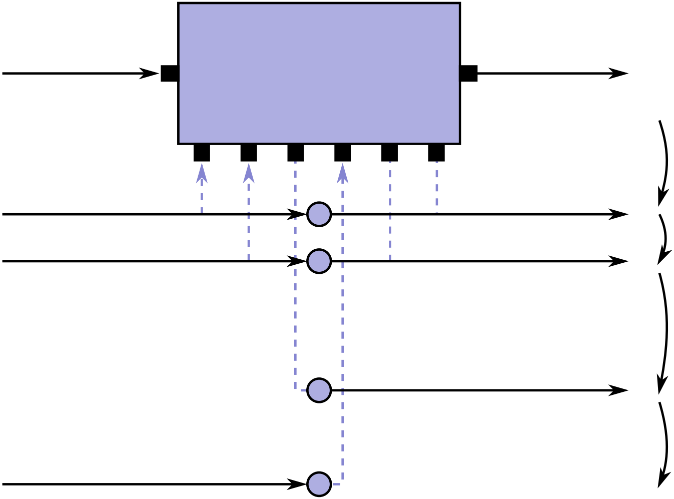
# Creating Dummy Nodes for Side Ports

## Local Optimization of Edge Crossings



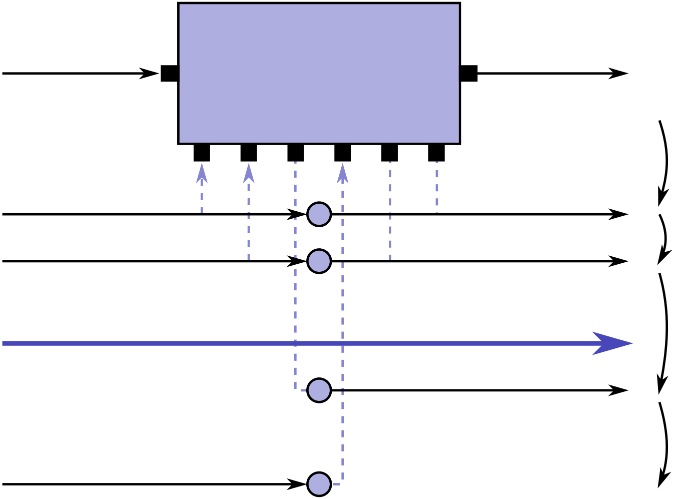
# Creating Dummy Nodes for Side Ports

## Local Optimization of Edge Crossings



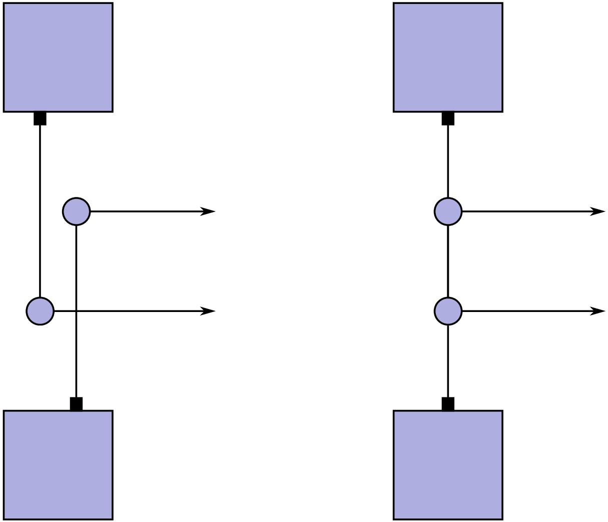
# Creating Dummy Nodes for Side Ports

## Local Optimization of Edge Crossings



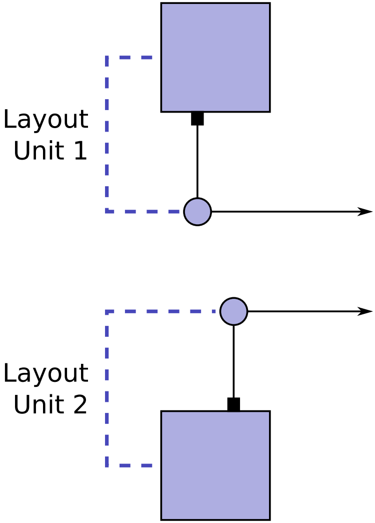
# Getting the Order of Dummy Nodes Right

## Introducing Layout Units to Avoid Ambiguity and Edge Crossings



# Getting the Order of Dummy Nodes Right

## Introducing Layout Units to Avoid Ambiguity and Edge Crossings

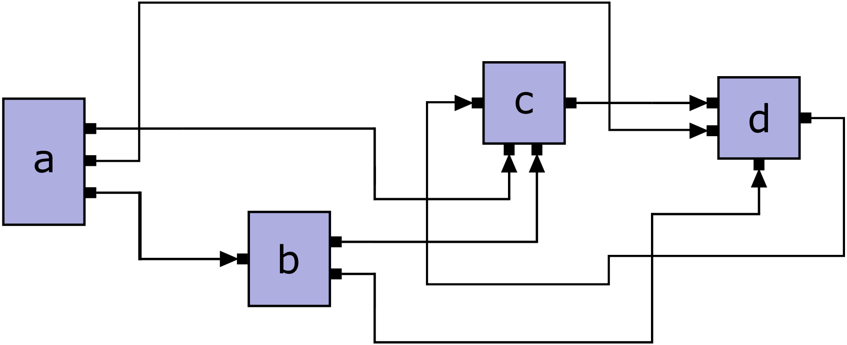


# Outline

1. The Importance of Port Support
   - Preliminaries
   - Inverted Ports
   - Side Ports
   - Putting It All Together
2. Demo Time!
3. Evaluation
4. Conclusion

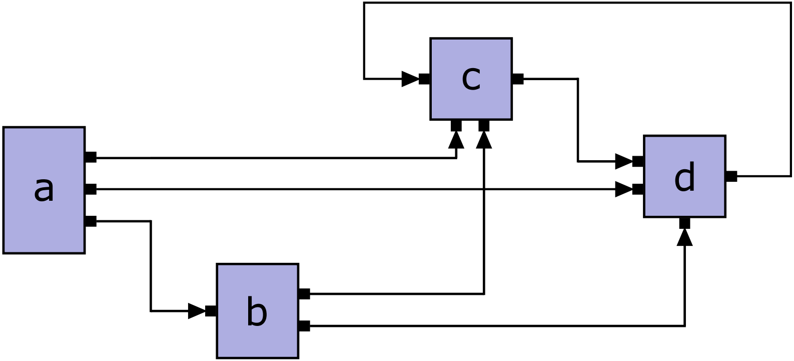
# The Diagram From the Beginning

## Layout Produced by Our Previous Algorithm



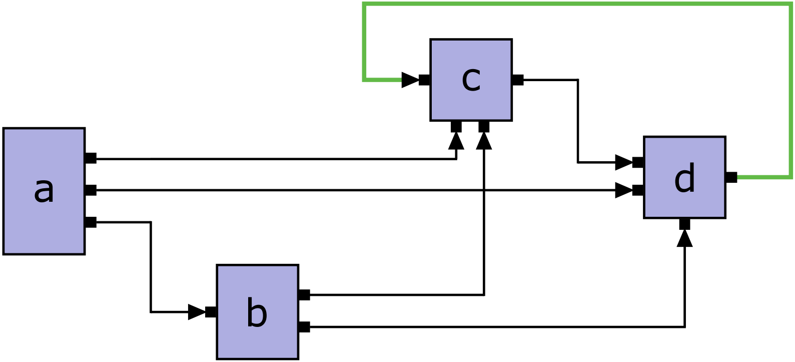
# The Diagram From the Beginning

## Layout Produced by Our Current Algorithm



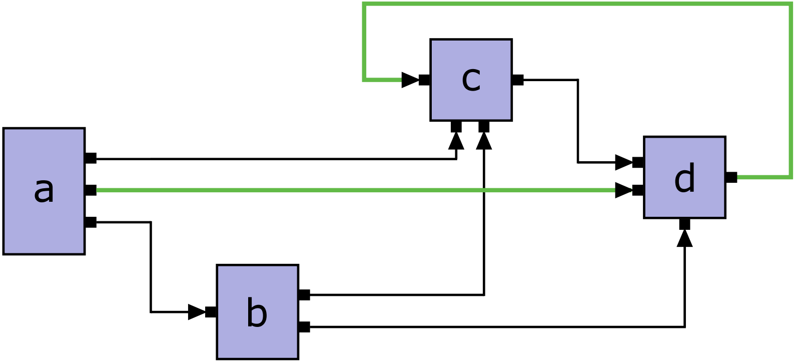
# The Diagram From the Beginning

## Layout Produced by Our Current Algorithm



# The Diagram From the Beginning

## Layout Produced by Our Current Algorithm



# Outline

1. The Importance of Port Support
    - Preliminaries
    - Inverted Ports
    - Side Ports
    - Putting It All Together
2. Demo Time!
3. Evaluation
4. Conclusion

# See the Algorithm in Action!

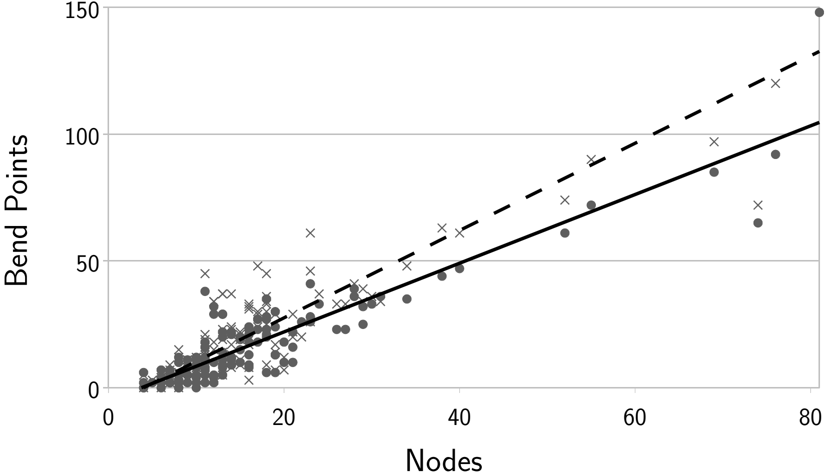
Please stand by for the KIELER tool to appear. . .

# Outline

1. The Importance of Port Support
   - Preliminaries
   - Inverted Ports
   - Side Ports
   - Putting It All Together
2. Demo Time!
3. Evaluation
4. Conclusion

# Layout Quality (Ptolemy Diagrams)

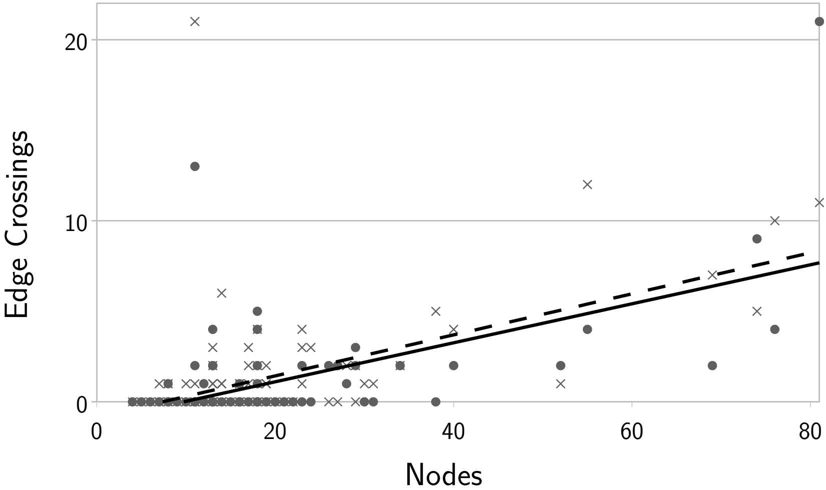
## Current Algorithm (solid) vs. Previous Algorithm (dashed)



141 Ptolemy demo models, up to 43 nodes per hierarchy level (avg. 8.98)

# Layout Quality (Ptolemy Diagrams)

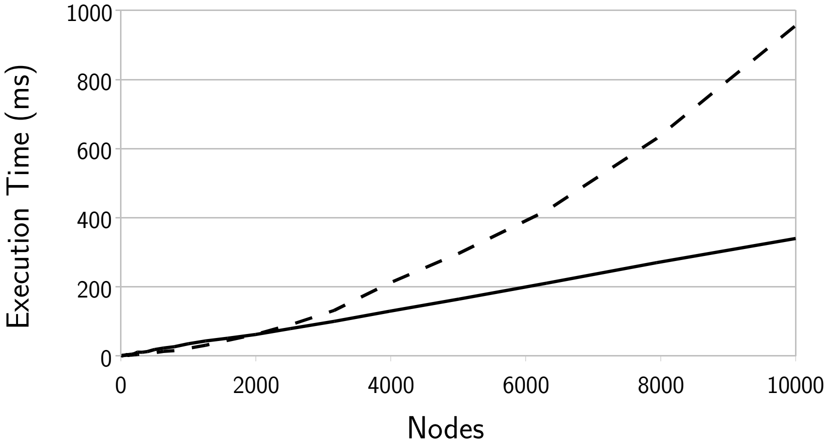
Current Algorithm (solid) vs. Previous Algorithm (dashed)



141 Ptolemy demo models, up to 43 nodes per hierarchy level (avg. 8.98)

# Runtime Performance

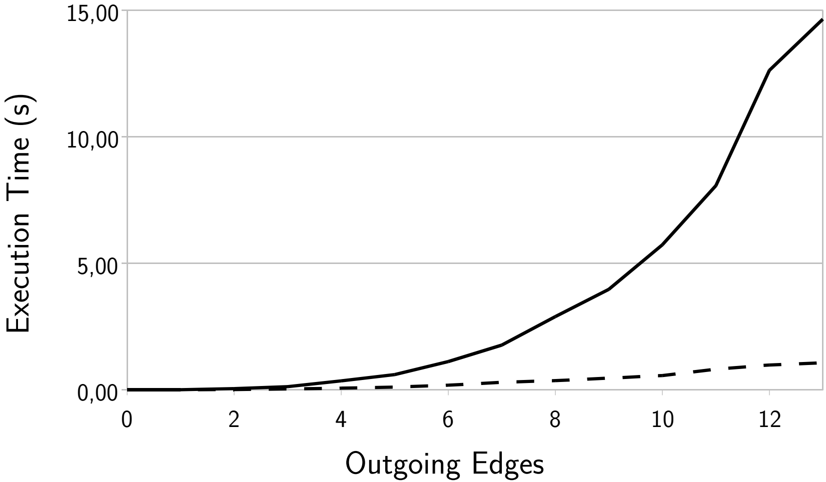
Current Algorithm (solid) vs. Previous Algorithm (dashed)



Random graphs with 10 to 10,000 nodes, 0 to 2 outgoing edges each

# Runtime Performance

## Current Algorithm (solid) vs. Previous Algorithm (dashed)



Random graphs with 100 nodes, fixed number of outgoing edges

# Outline

1. The Importance of Port Support
   - Preliminaries
   - Inverted Ports
   - Side Ports
   - Putting It All Together
2. Demo Time!
3. Evaluation
4. Conclusion

# Summary

## What Did Just Happen?

- What we talked about:
    - More routing flexibility through dummy nodes
    - Demonstrated with inverted ports and side ports

# Summary

## What Did Just Happen?

- What we talked about:
  - More routing flexibility through dummy nodes
  - Demonstrated with inverted ports and side ports
- What we didn't talk about:
  - Ports in nested diagrams
  - Flexible architectures for layout algorithms

# Summary

## What Did Just Happen?

- What we talked about:
  - More routing flexibility through dummy nodes
  - Demonstrated with inverted ports and side ports
- What we didn’t talk about:
  - Ports in nested diagrams
  - Flexible architectures for layout algorithms
- Examples of what still remains to be done:
  - Label placement
  - Better node placement to avoid unnecessary bend points
  - Multithreading in layout algorithms

# Summary

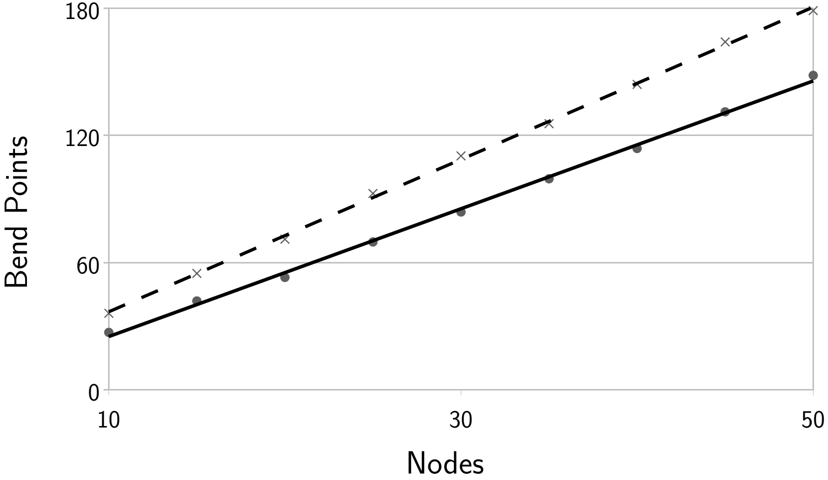
## What Did Just Happen?

- What we talked about:
  - More routing flexibility through dummy nodes
  - Demonstrated with inverted ports and side ports
- What we didn't talk about:
  - Ports in nested diagrams
  - Flexible architectures for layout algorithms
- Examples of what still remains to be done:
  - Label placement
  - Better node placement to avoid unnecessary bend points
  - Multithreading in layout algorithms

Feel free to pay us a visit and look at our KIELER platform at
`www.informatik.uni-kiel.de/rtsys`

# Layout Quality (Random Diagrams)

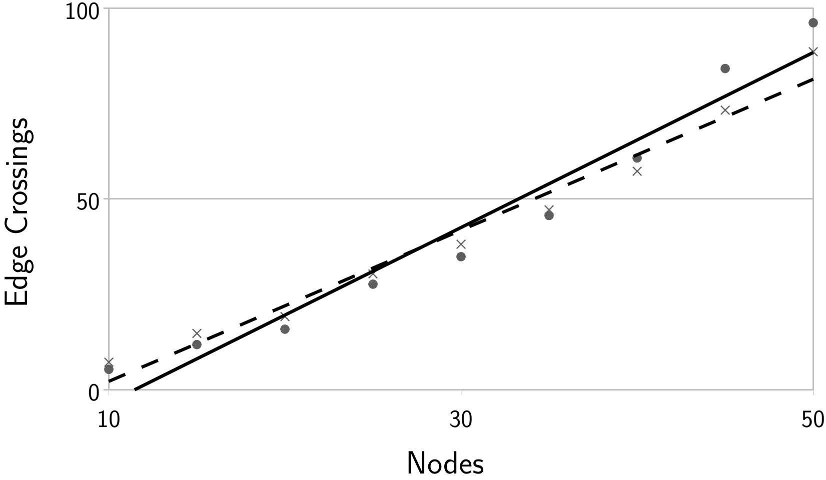
Current Algorithm (solid) vs. Previous Algorithm (dashed)



270 random graphs, 10 to 50 nodes, avg. 1.2 outgoing edges per node

# Layout Quality (Random Diagrams)

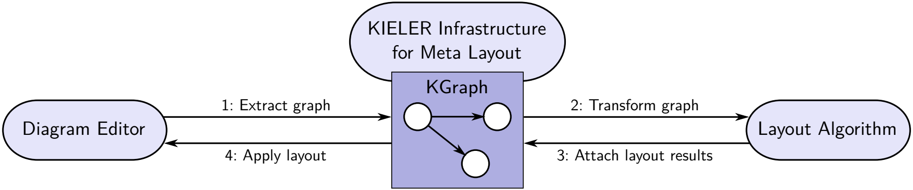
Current Algorithm (solid) vs. Previous Algorithm (dashed)



270 random graphs, 10 to 50 nodes, avg. 1.2 outgoing edges per node

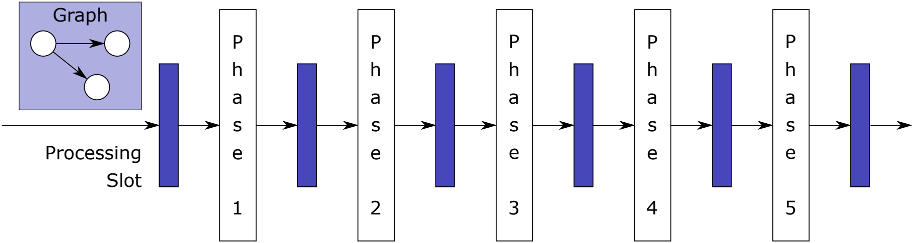
# Connecting Editors and Layout Algorithms

## The KIELER Infrastructure for Meta Layout



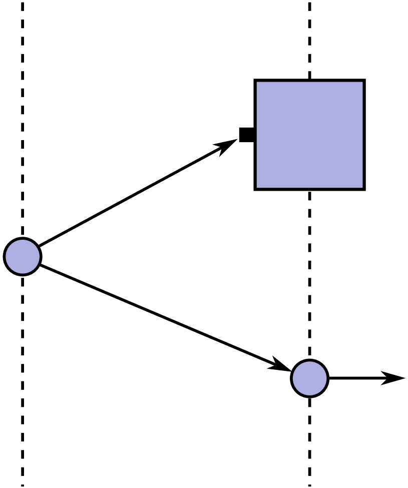
# The Algorithm’s Architecture

## Interchangeable Phases and Intermediate Processors



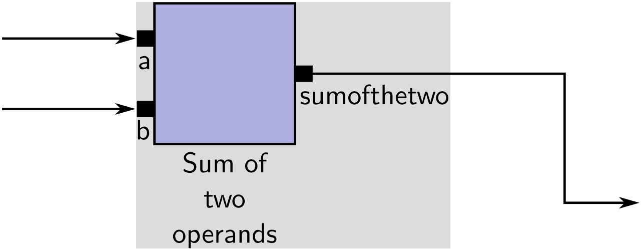
# Inverted Ports

A Solution with Two Dummy Nodes



# Basic Support for Node and Port Labels

## Reserving Space through Bounding Boxes


a
b
sumofthetwo
Sum of
two
operands