

Sabotage Game and Sabotage Modal Logic

GAMES 2003 – Vienna

Philipp Rohde

joint work with Christof Löding

`rohde@informatik.rwth-aachen.de`

Lehrstuhl für Informatik VII

RWTH Aachen

Motivation

Reactive systems with dynamic changes of the system itself

Motivation

Reactive systems with dynamic changes of the system itself

Two processes:

- **Local** movement within the system
- **Global** change of the system

Motivation

Reactive systems with dynamic changes of the system itself

Two processes:

- **Local** movement within the system
- **Global** change of the system

Here: removing objects

Motivation

Reactive systems with dynamic changes of the system itself

Two processes:

- **Local** movement within the system
- **Global** change of the system

Here: removing objects

Example: Server networks where servers stop working or connections break down (malfunction or damage)

Motivation

Reactive systems with dynamic changes of the system itself

Two processes:

- **Local** movement within the system
- **Global** change of the system

Here: removing objects

Example: Server networks where servers stop working or connections break down (malfunction or damage)

Real travelling salesman: Find a way between cities within a railway network where someone (?) starts cancelling connections

The sabotage game (van Benthem, 2002)

Game arena:

Undirected, finite multi-graph; initial vertex; final vertex

The sabotage game (van Benthem, 2002)

Game arena:

Undirected, finite multi-graph; initial vertex; final vertex

Two player game:

Runner and Blocker; Runner starts at initial vertex

The sabotage game (van Benthem, 2002)

Game arena:

Undirected, finite multi-graph; initial vertex; final vertex

Two player game:

Runner and Blocker; Runner starts at initial vertex

A game round:

1. Runner moves along existing edge
2. Blocker deletes an edge (somewhere in the graph)

The sabotage game (van Benthem, 2002)

Game arena:

Undirected, finite multi-graph; initial vertex; final vertex

Two player game:

Runner and Blocker; Runner starts at initial vertex

A game round:

1. Runner moves along existing edge
2. Blocker deletes an edge (somewhere in the graph)

End of game:

If Runner cannot make a move (he loses) **or**

Runner reaches final vertex (he wins)

The sabotage game (van Benthem, 2002)

Game arena:

Undirected, finite multi-graph; initial vertex; final vertex

Two player game:

Runner and Blocker; Runner starts at initial vertex

A game round:

1. Runner moves along existing edge
2. Blocker deletes an edge (somewhere in the graph)

End of game:

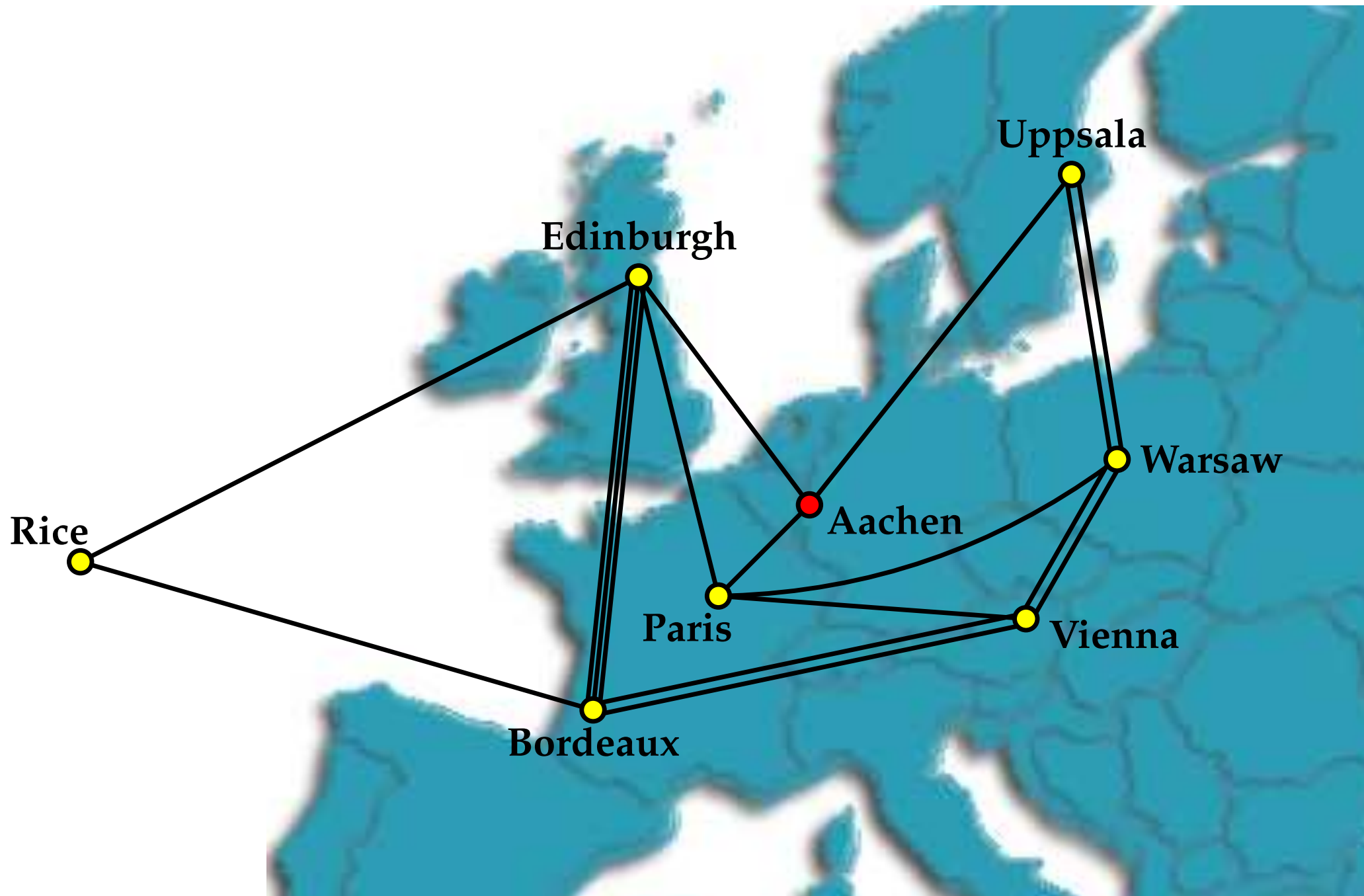
If Runner cannot make a move (he loses) **or**

Runner reaches final vertex (he wins)

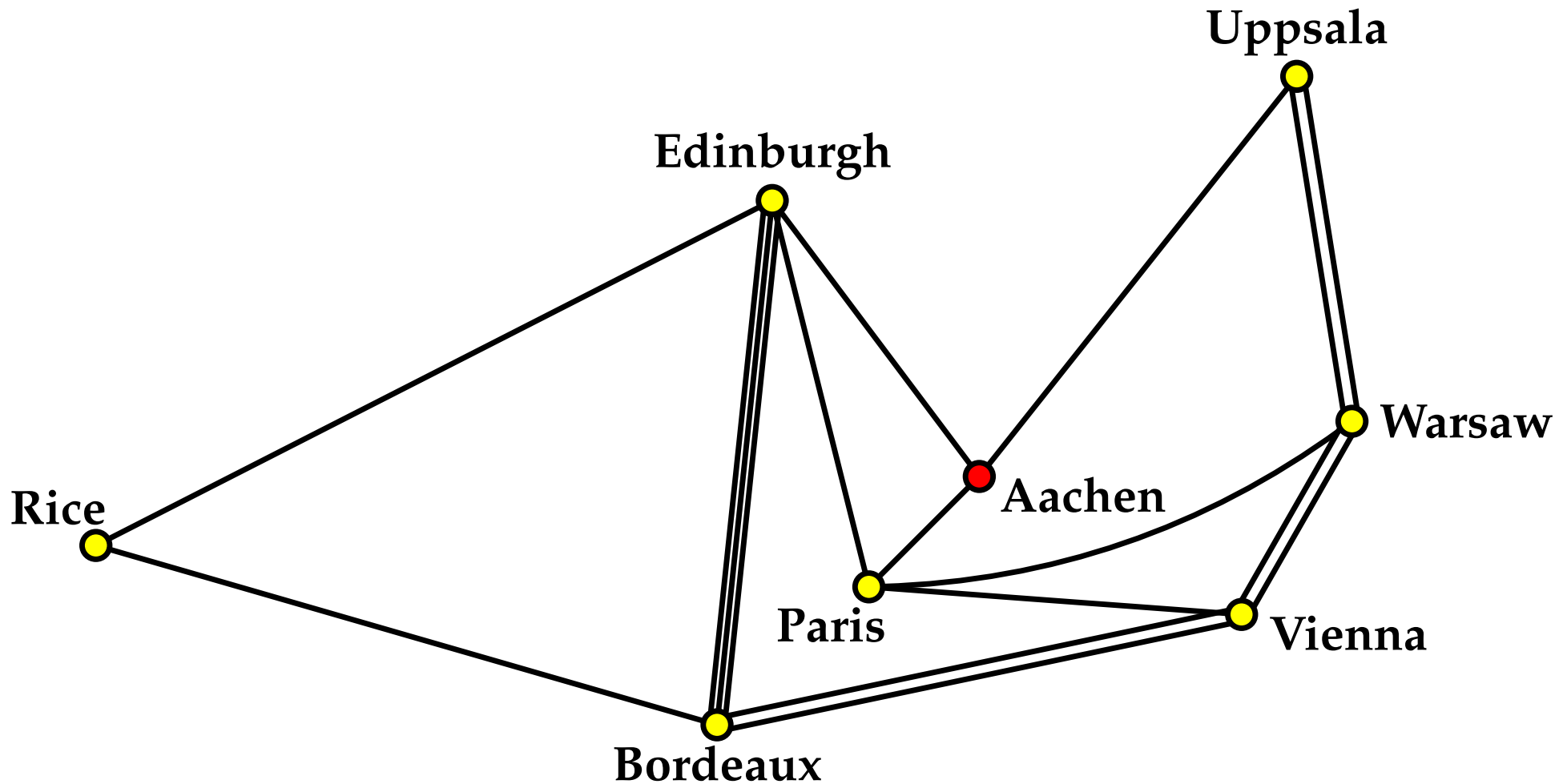
Game is finite:

rounds \leq # edges (with multiplicity)

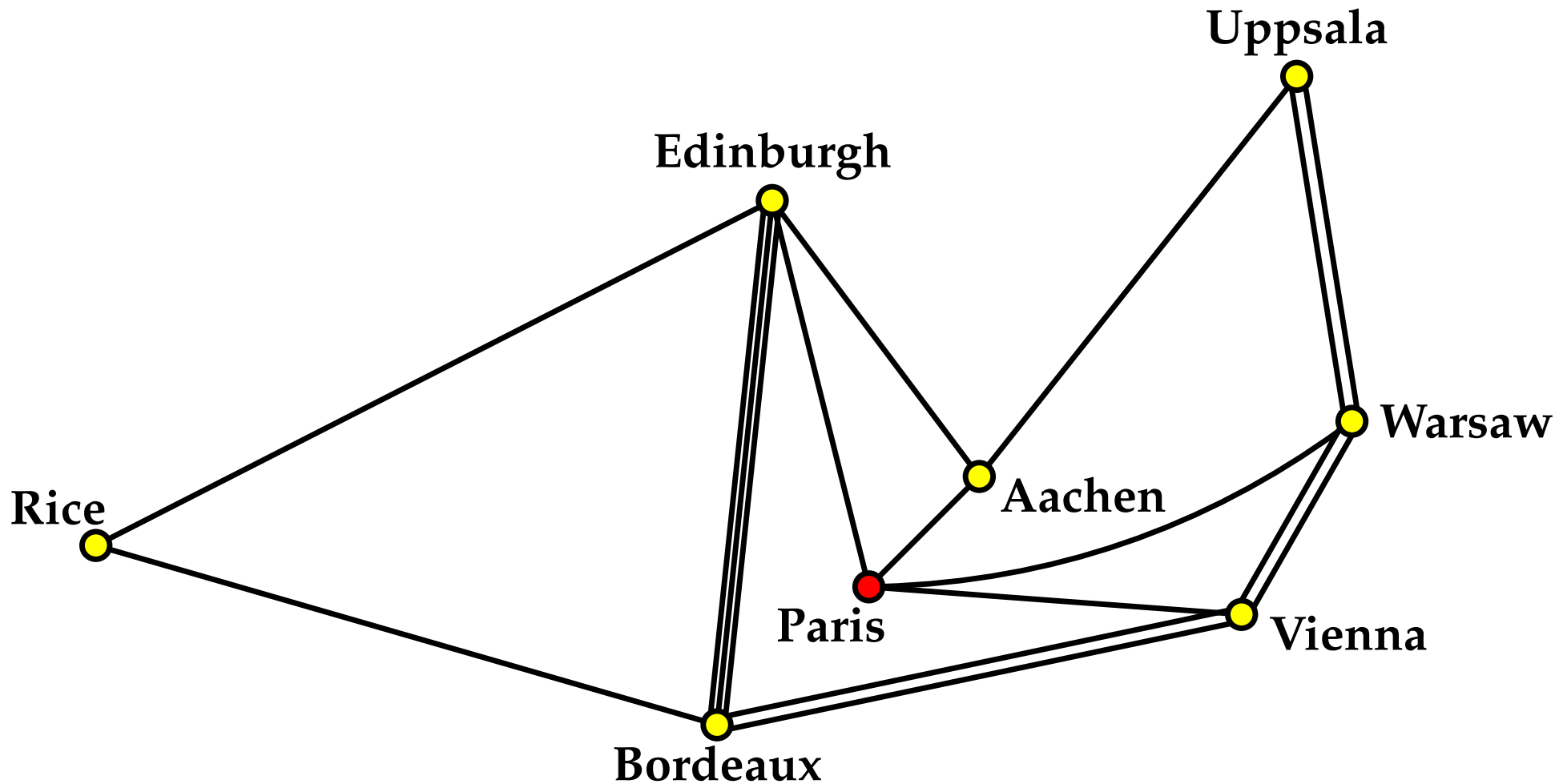
Example: From Aachen to Vienna



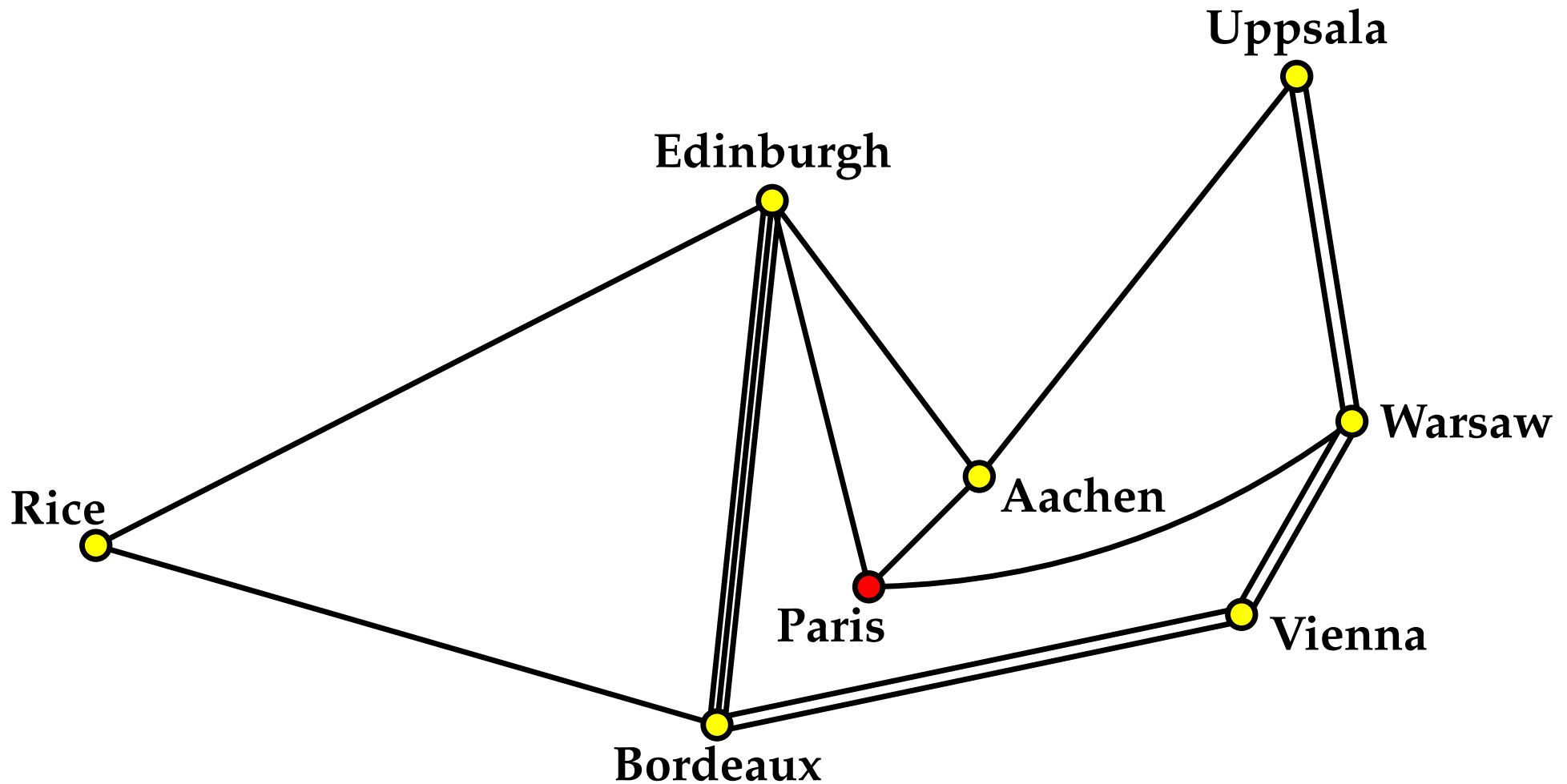
Example: From Aachen to Vienna



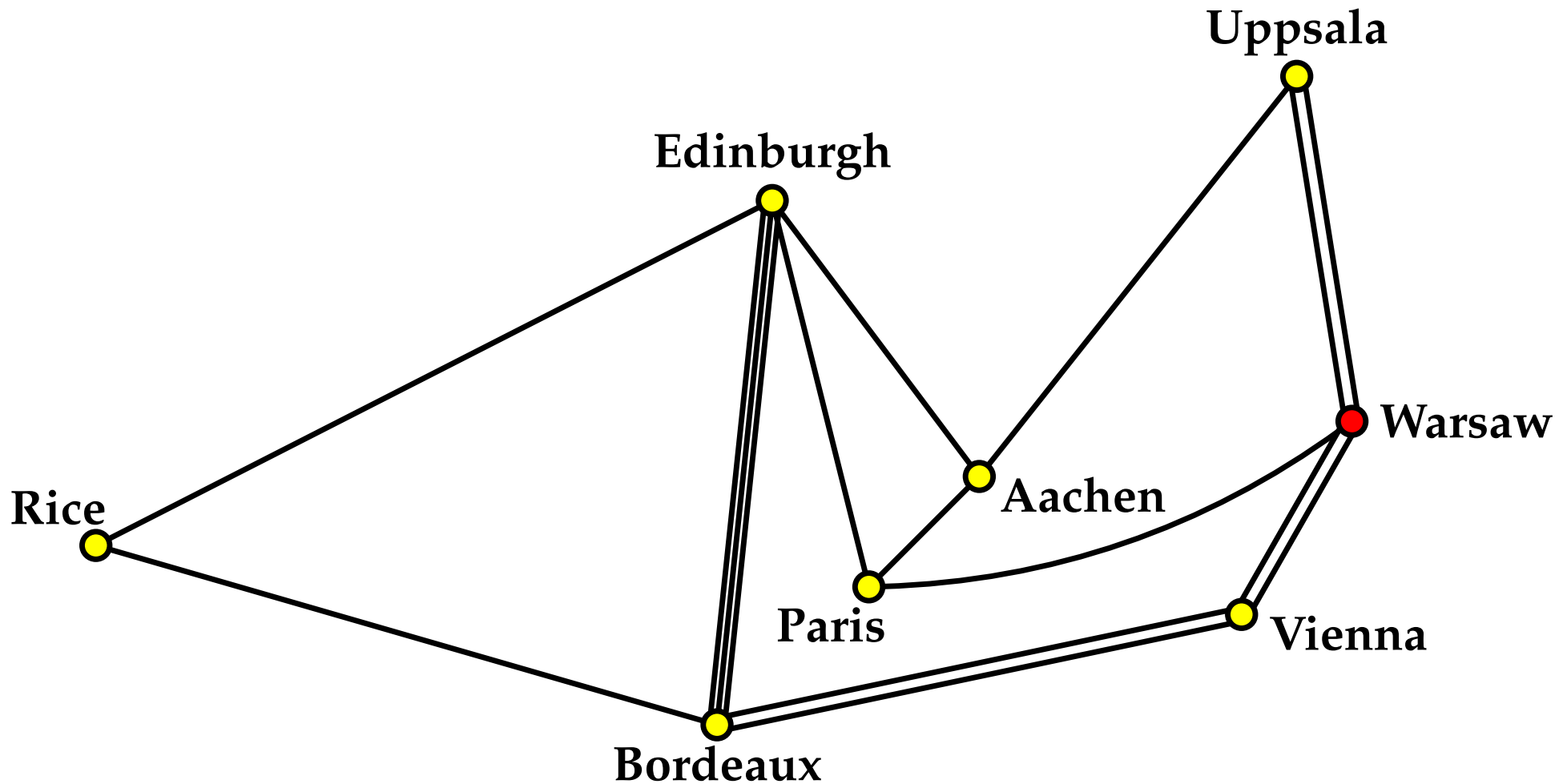
Example: From Aachen to Vienna



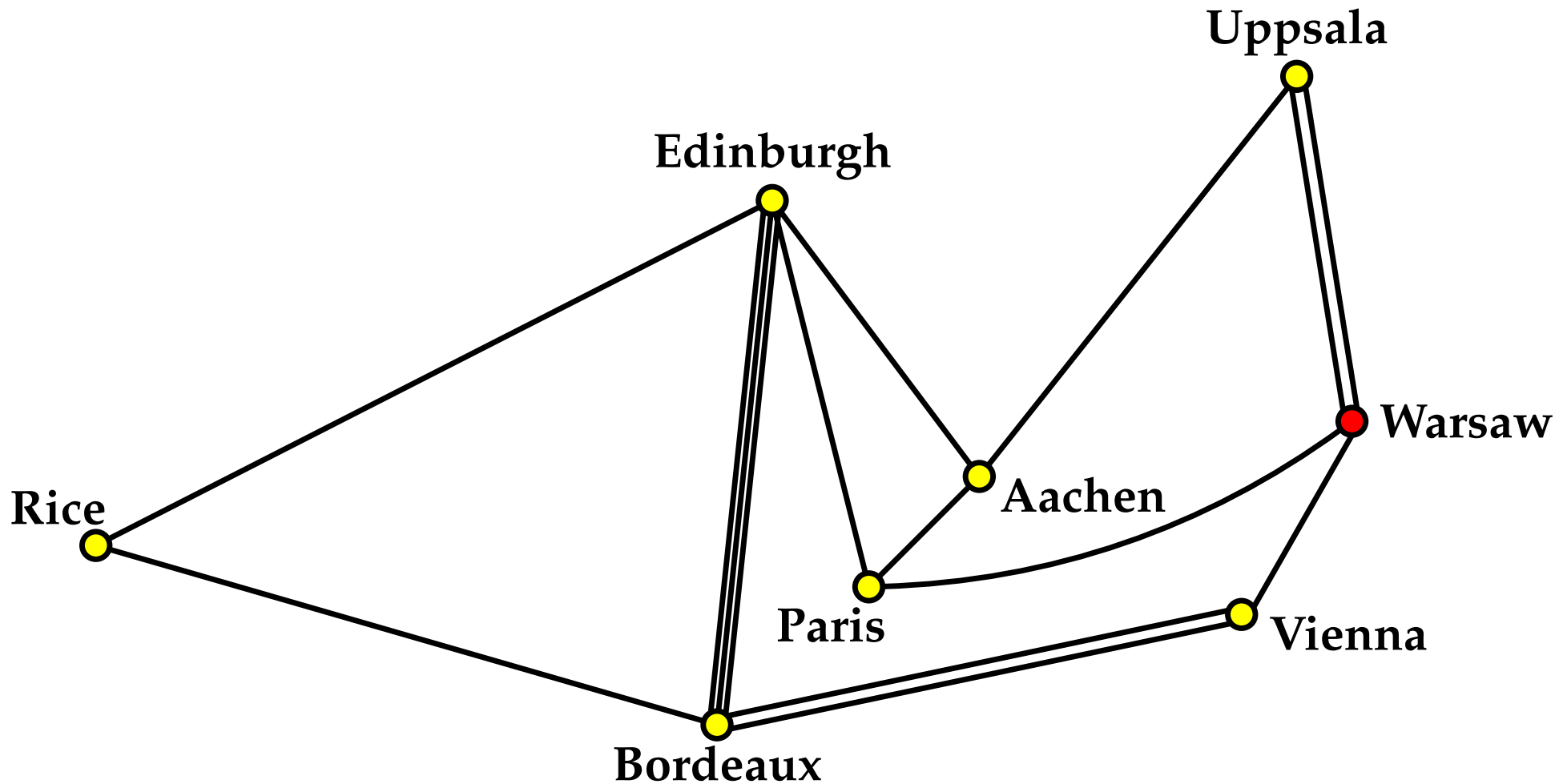
Example: From Aachen to Vienna



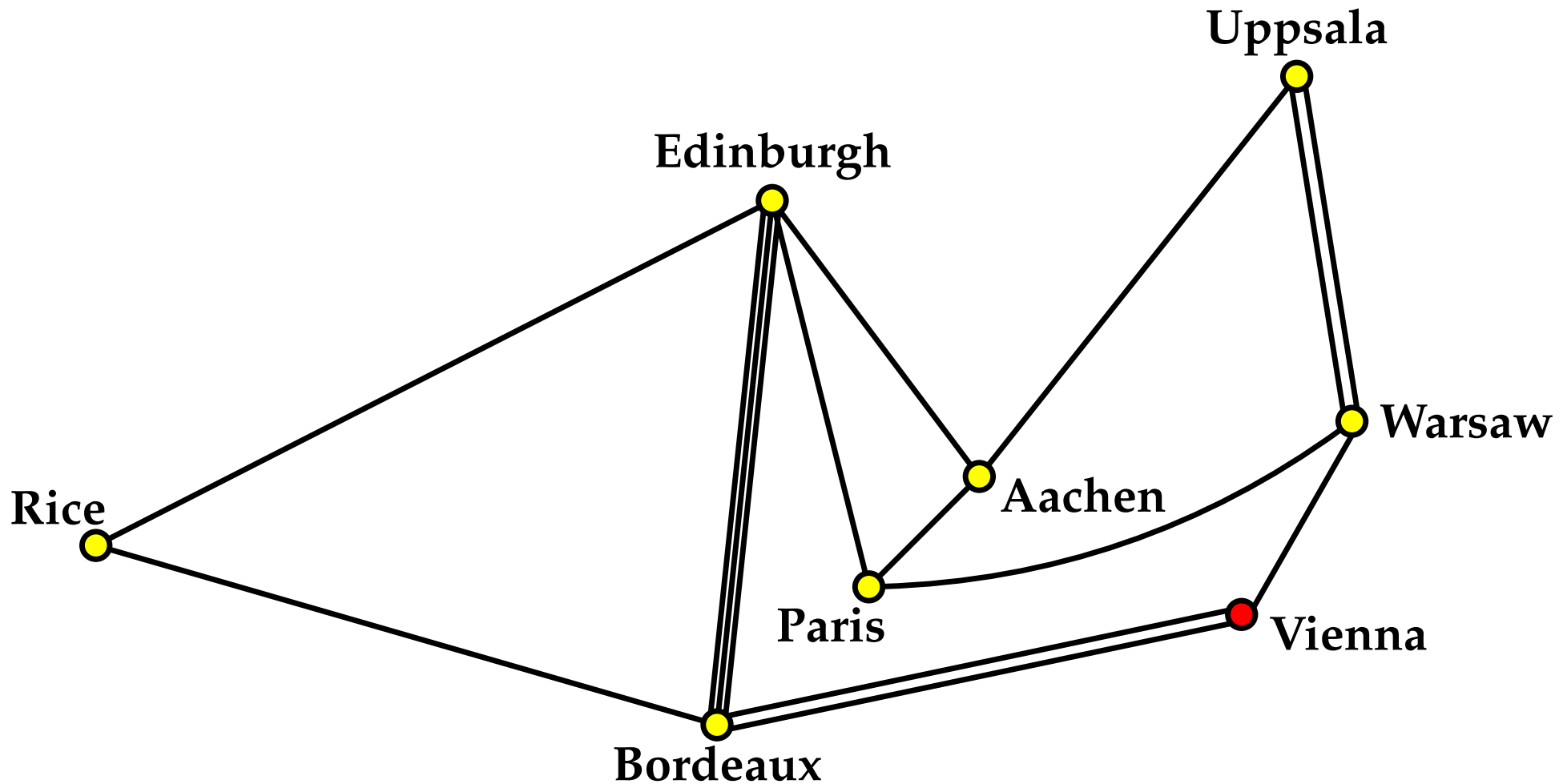
Example: From Aachen to Vienna



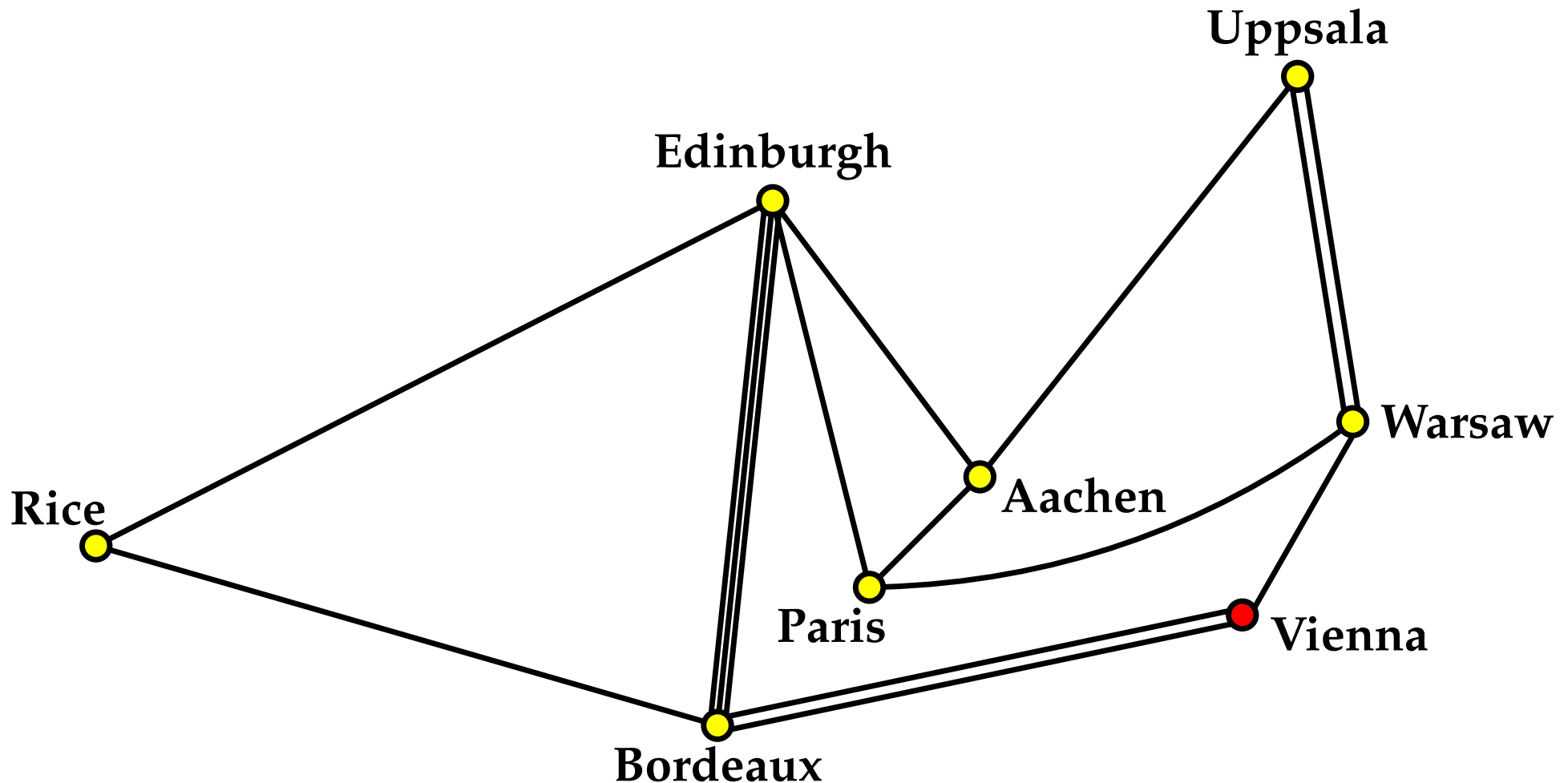
Example: From Aachen to Vienna



Example: From Aachen to Vienna



Example: From Aachen to Vienna



Main aspect: Runner moves **locally**, Blocker acts **globally**

Solving the sabotage game is PSPACE-complete

Question: Who has a winning strategy?

Solving the sabotage game is PSPACE-complete

Question: Who has a winning strategy?

In PSPACE: Compute all possible games (finitely many).

Storing positions and deleted edges takes space polynomial in size of game graph (# vertices + # edges with multiplicity).

Solving the sabotage game is PSPACE-complete

Question: Who has a winning strategy?

In PSPACE: Compute all possible games (finitely many).

Storing positions and deleted edges takes space polynomial in size of game graph (# vertices + # edges with multiplicity).

Alternatively: Game description as FO-formula and FO model checking.

Solving the sabotage game is PSPACE-complete

Question: Who has a winning strategy?

In PSPACE: Compute all possible games (finitely many).

Storing positions and deleted edges takes space polynomial in size of game graph (# vertices + # edges with multiplicity).

Alternatively: Game description as FO-formula and FO model checking.

Theorem (MFCS 2003): There is a polynomial time reduction from **Quantified Boolean Formulas (QBF)** to sabotage games.

Solving the sabotage game is PSPACE-complete

Question: Who has a winning strategy?

In PSPACE: Compute all possible games (finitely many).

Storing positions and deleted edges takes space polynomial in size of game graph (# vertices + # edges with multiplicity).

Alternatively: Game description as FO-formula and FO model checking.

Theorem (MFCS 2003): There is a polynomial time reduction from **Quantified Boolean Formulas (QBF)** to sabotage games.

In particular solving these games is PSPACE-hard.

PSPACE-completeness for variants of the game

Other winning conditions:

PSPACE-completeness for variants of the game

Other winning conditions:

- **Complete search:**
Runner wins iff each vertex is visited

PSPACE-completeness for variants of the game

Other winning conditions:

- **Complete search:**
Runner wins iff each vertex is visited
- **Hamilton path:**
Runner wins iff each vertex is visited exactly once

PSPACE-completeness for variants of the game

Other winning conditions:

- **Complete search:**
Runner wins iff each vertex is visited
- **Hamilton path:**
Runner wins iff each vertex is visited exactly once

Other game rules:

- **Blocker removes up to n edges in each round, n fixed**

PSPACE-completeness for variants of the game

Other winning conditions:

- **Complete search:**
Runner wins iff each vertex is visited
- **Hamilton path:**
Runner wins iff each vertex is visited exactly once

Other game rules:

- Blocker removes up to n edges in each round, n fixed
- Blocker removes one vertex in each round (together with connected edges)
Restriction: not the current vertex, not the final vertex

PSPACE-completeness for variants of the game

Other winning conditions:

- Complete search:
Runner wins iff each vertex is visited
- Hamilton path:
Runner wins iff each vertex is visited exactly once

Other game rules:

- Blocker removes up to n edges in each round, n fixed
- Blocker removes one vertex in each round (together with connected edges)
Restriction: not the current vertex, not the final vertex
- Blocker removes up to n vertices in each round, n fixed

Sabotage modal logic SML

Modal logic over **changing models**: modality for deleting transitions.

Interpreted over labelled transition systems \mathcal{T} (alphabet Σ).

Sabotage modal logic SML

Modal logic over **changing models**: modality for deleting transitions.

Interpreted over labelled transition systems \mathcal{T} (alphabet Σ).

Syntax of SML:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \diamond_a\varphi \mid \blacklozenge_a\varphi$$

(p predicate symbol, $a \in \Sigma$ transition label)

Sabotage modal logic SML

Modal logic over **changing models**: modality for deleting transitions.

Interpreted over labelled transition systems \mathcal{T} (alphabet Σ).

Syntax of SML:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \diamond_a\varphi \mid \blacklozenge_a\varphi$$

(p predicate symbol, $a \in \Sigma$ transition label)

'Sabotage box' defined as $\boxminus_a\varphi ::= \neg\blacklozenge_a\neg\varphi$

Sabotage modal logic SML

Modal logic over **changing models**: modality for deleting transitions.

Interpreted over labelled transition systems \mathcal{T} (alphabet Σ).

Syntax of SML:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \diamond_a\varphi \mid \blacklozenge_a\varphi$$

(p predicate symbol, $a \in \Sigma$ transition label)

'Sabotage box' defined as $\boxplus_a\varphi ::= \neg\blacklozenge_a\neg\varphi$

Semantics of SML: Usual modal logic +

$\langle \mathcal{T}, s \rangle \models \blacklozenge_a\varphi$ iff there is a -transition t of \mathcal{T} such that
 $\langle \mathcal{T} \setminus t, s \rangle \models \varphi$

Model checking and model-theoretic properties

SML can be embedded into FO \rightsquigarrow **model checking** in PSPACE

Sabotage game \rightsquigarrow **model checking** is PSPACE-complete

Model checking and model-theoretic properties

SML can be embedded into FO \rightsquigarrow **model checking** in PSPACE

Sabotage game \rightsquigarrow **model checking** is PSPACE-complete

Formula complexity is linear; **program complexity** is polynomial

Model checking and model-theoretic properties

SML can be embedded into FO \rightsquigarrow **model checking** in PSPACE

Sabotage game \rightsquigarrow **model checking** is PSPACE-complete

Formula complexity is linear; **program complexity** is polynomial

SML versus modal logic (FSTTCS 2003):

Model checking and model-theoretic properties

SML can be embedded into FO \rightsquigarrow **model checking** in PSPACE

Sabotage game \rightsquigarrow **model checking** is PSPACE-complete

Formula complexity is linear; **program complexity** is polynomial

SML versus modal logic (FSTTCS 2003):

- SML can fix the **number of successors**

Model checking and model-theoretic properties

SML can be embedded into FO \rightsquigarrow **model checking** in PSPACE

Sabotage game \rightsquigarrow **model checking** is PSPACE-complete

Formula complexity is linear; **program complexity** is polynomial

SML versus modal logic (FSTTCS 2003):

- SML can fix the **number of successors**
- SML does not have the **tree model property**, hence it is not **bisimulation-invariant**

Model checking and model-theoretic properties

SML can be embedded into FO \rightsquigarrow **model checking** in PSPACE

Sabotage game \rightsquigarrow **model checking** is PSPACE-complete

Formula complexity is linear; **program complexity** is polynomial

SML versus modal logic (FSTTCS 2003):

- SML can fix the **number of successors**
- SML does not have the **tree model property**, hence it is not **bisimulation-invariant**
- SML does not have the **finite model property**

Example and undecidability results

Example: $\varphi \equiv \diamond\diamond\top \wedge \exists\Box\perp$

Example and undecidability results

Example: $\varphi \equiv \Diamond\Diamond T \wedge \Box\Box\perp$

$\Diamond\Diamond T \rightsquigarrow$ “you can go two steps further”

$\Box\Box\perp \rightsquigarrow$ “no matter which transition is deleted,
you cannot go any step further”

Example and undecidability results

Example: $\varphi \equiv \diamond\diamond\top \wedge \Box\Box\perp$

$\diamond\diamond\top \rightsquigarrow$ “you can go two steps further”

$\Box\Box\perp \rightsquigarrow$ “no matter which transition is deleted,
you cannot go any step further”


 is not a model

Example and undecidability results

Example: $\varphi \equiv \Diamond\Diamond T \wedge \Box\Box\perp$

$\Diamond\Diamond T \rightsquigarrow$ “you can go two steps further”

$\Box\Box\perp \rightsquigarrow$ “no matter which transition is deleted,
you cannot go any step further”

 is not a model

Example and undecidability results

Example: $\varphi \equiv \diamond\diamond\top \wedge \Box\Box\perp$

$\diamond\diamond\top \rightsquigarrow$ “you can go two steps further”

$\Box\Box\perp \rightsquigarrow$ “no matter which transition is deleted,
you cannot go any step further”



is the only model (except for isolated states)

Example and undecidability results

Example: $\varphi \equiv \diamond\diamond\top \wedge \Box\Box\perp$

$\diamond\diamond\top \rightsquigarrow$ “you can go two steps further”

$\Box\Box\perp \rightsquigarrow$ “no matter which transition is deleted,
you cannot go any step further”



is the only model (except for isolated states)

Theorem (FSTTCS 2003): The decision problems

- Satisfiability
- Finite Satisfiability
- Infinity Axiom

for SML are undecidable.

Example and undecidability results

Example: $\varphi \equiv \diamond\diamond\top \wedge \Box\Box\perp$

$\diamond\diamond\top \rightsquigarrow$ “you can go two steps further”

$\Box\Box\perp \rightsquigarrow$ “no matter which transition is deleted,
you cannot go any step further”



is the only model (except for isolated states)

Theorem (FSTTCS 2003): The decision problems

- Satisfiability
- Finite Satisfiability
- Infinity Axiom

for SML are undecidable.

Proof by encoding variants of **Post's Correspondence Problem**.

Sabotage operator strengthen modal logic essentially:

Summary / Outlook

**Sabotage operator strengthen modal logic essentially:
SML much more resembles FO than usual modal logic,**

Summary / Outlook

**Sabotage operator strengthen modal logic essentially:
SML much more resembles FO than usual modal logic,
but SML has lower formula complexity.**

Summary / Outlook

Sabotage operator strengthen modal logic essentially:

SML much more resembles FO than usual modal logic,

but SML has lower formula complexity.

Relations to dynamic-epistemic logics / logics of knowledge.

Summary / Outlook

Sabotage operator strengthen modal logic essentially:

SML much more resembles FO than usual modal logic,
but SML has lower formula complexity.

Relations to dynamic-epistemic logics / logics of knowledge.

Open questions:

- Infinite game graphs? Infinite versions of the game?

Summary / Outlook

Sabotage operator strengthen modal logic essentially:

**SML much more resembles FO than usual modal logic,
but SML has lower formula complexity.**

Relations to dynamic-epistemic logics / logics of knowledge.

Open questions:

- **Infinite game graphs? Infinite versions of the game?**
- **Restrictions to the global power of sabotage operator? E.g., deleted edges have bounded distance to current position?**

Summary / Outlook

Sabotage operator strengthen modal logic essentially:

SML much more resembles FO than usual modal logic,
but SML has lower formula complexity.

Relations to dynamic-epistemic logics / logics of knowledge.

Open questions:

- Infinite game graphs? Infinite versions of the game?
- Restrictions to the global power of sabotage operator? E.g., deleted edges have bounded distance to current position?
- Characteristic notion of bisimulation for SML?

Summary / Outlook

Sabotage operator strengthen modal logic essentially:

**SML much more resembles FO than usual modal logic,
but SML has lower formula complexity.**

Relations to dynamic-epistemic logics / logics of knowledge.

Open questions:

- **Infinite game graphs? Infinite versions of the game?**
- **Restrictions to the global power of sabotage operator? E.g., deleted edges have bounded distance to current position?**
- **Characteristic notion of bisimulation for SML?**
- **Valid principles for SML? Interaction laws between modalities?**