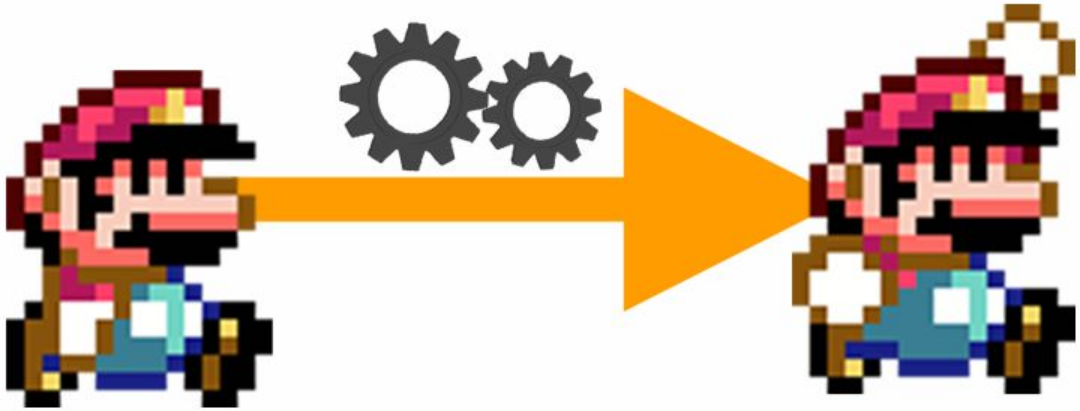
Operational Transformation

and its Relevance in Games



by Felix Pauck, Patrick Steffens

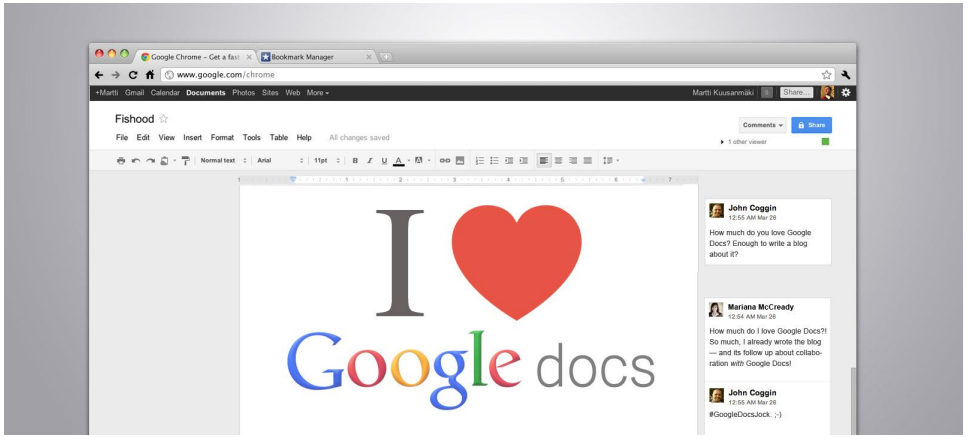
Outline

- **Introduction**
- **Solitaire**
  - Simplified Solitaire
  - Execution Scenario
- **Consistency Models**
  - CC Model
  - CCI Model
- **OT Algorithm**
  - OT Functions
- **Collaborative Games**

Introduction

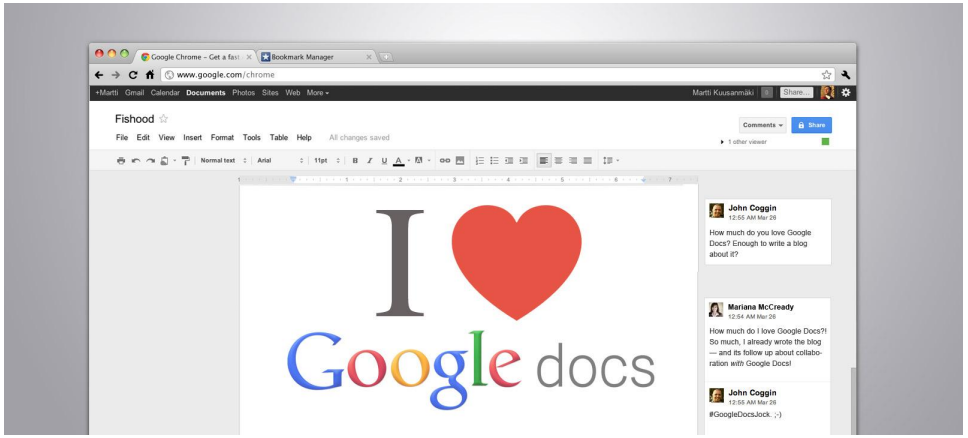
Google Docs

- Collaborative real-time editor
- Effects of operation instantly available



Introduction

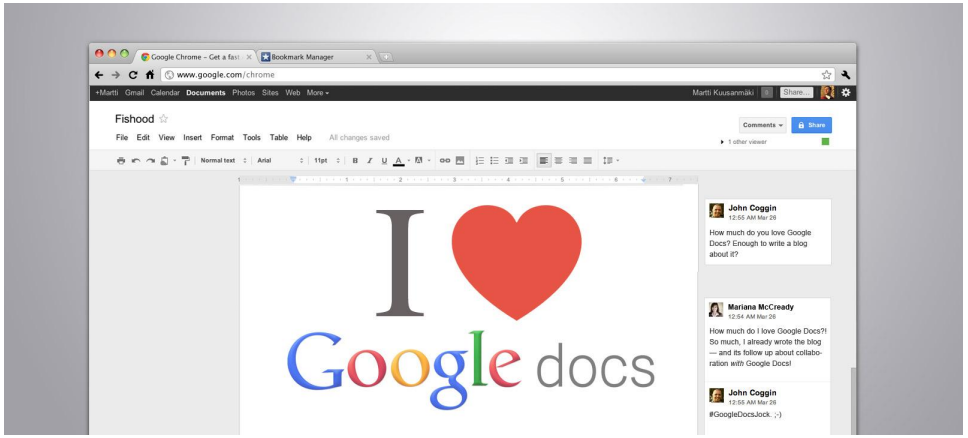
Google Docs



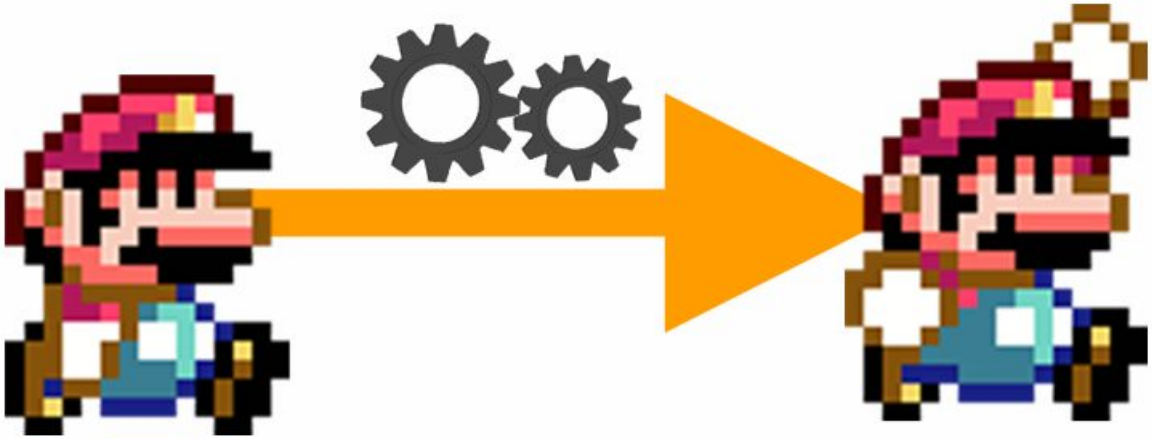
- Collaborative real-time editor
- Effects of operation instantly available
- Conflicts can occur
- Solution: **Operational Transformation**

Introduction

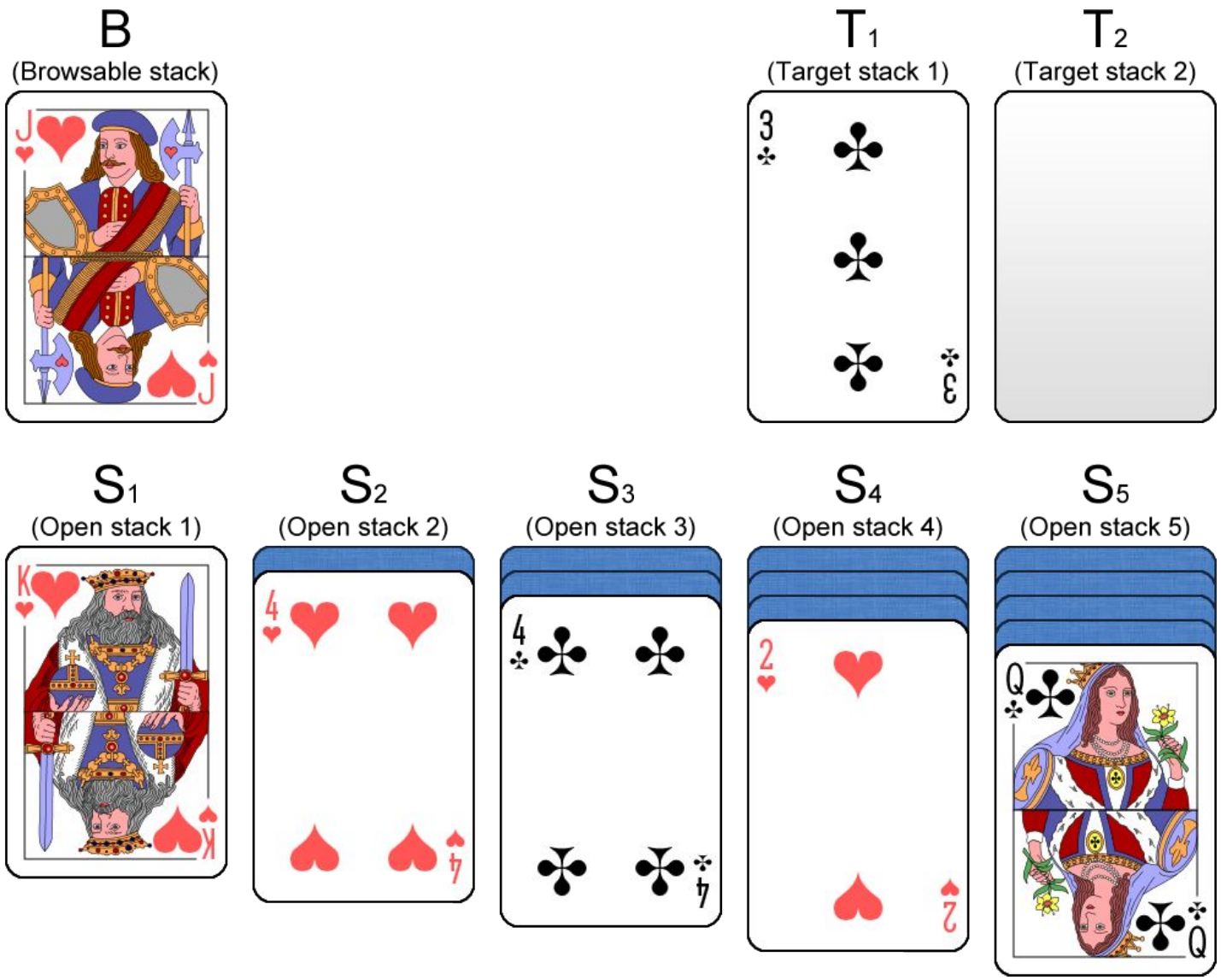
Google Docs



- Collaborative real-time editor
- Effects of operation instantly available
- Conflicts can occur
- Solution: **Operational Transformation**



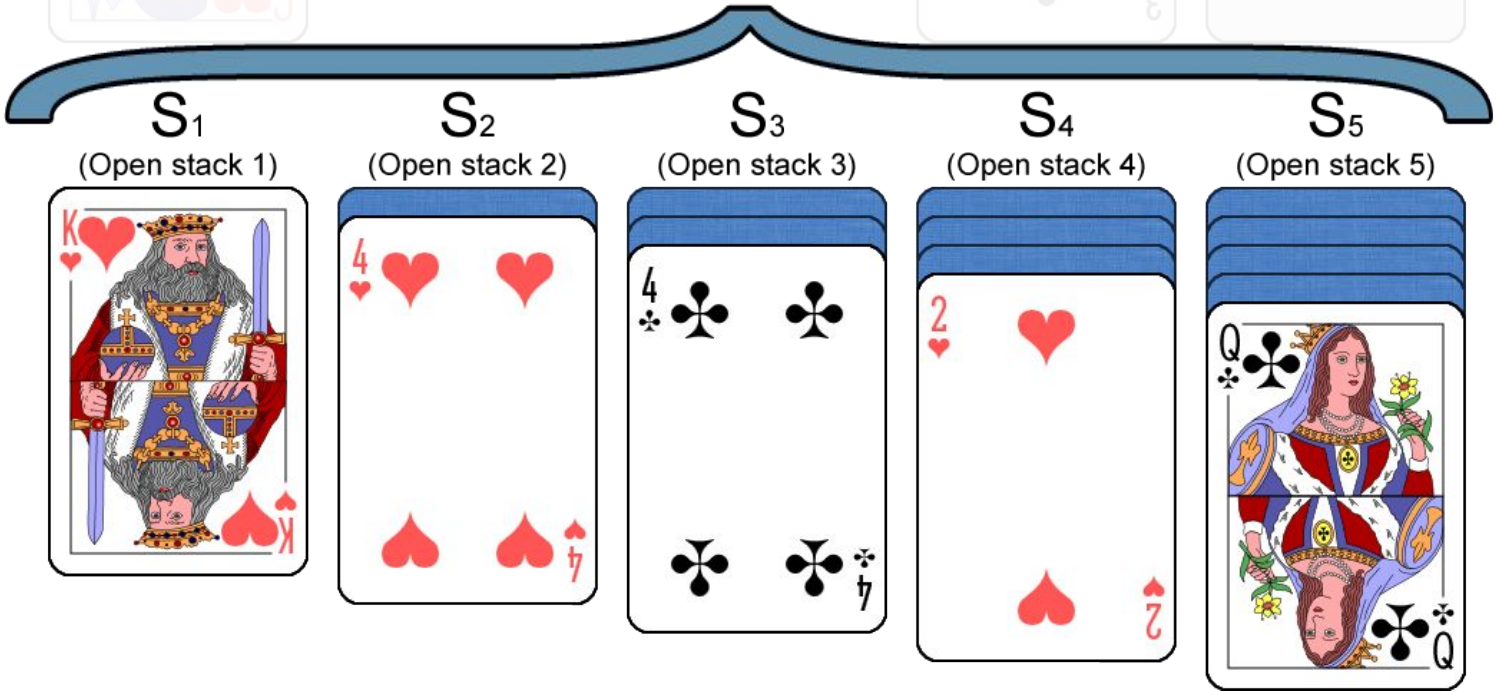
Simplified Solitaire



Simplified Solitaire

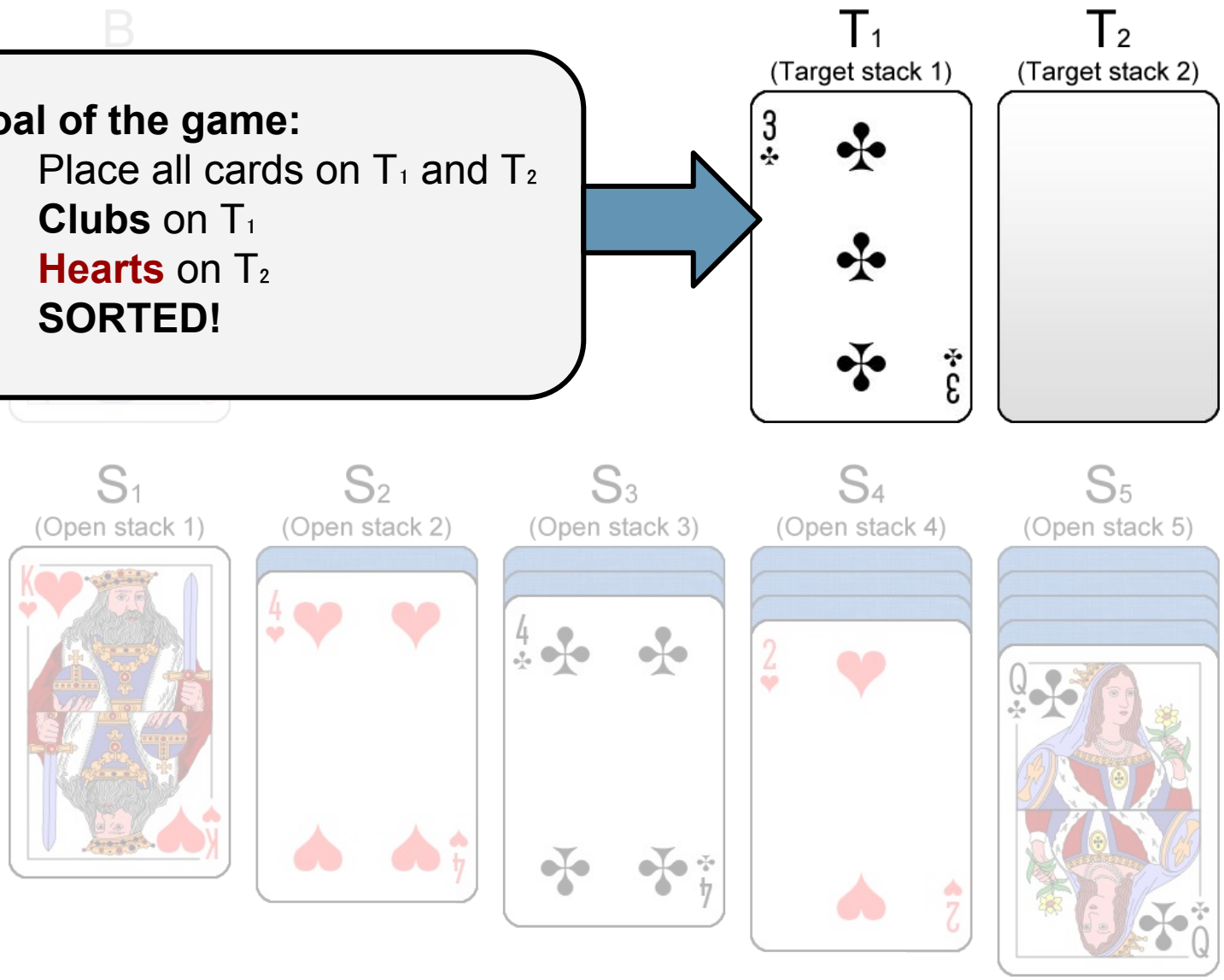
- **Clubs** on **Hearts**
- **Hearts** on **Clubs**

- Hidden cards
  (can only be uncovered if they are on top)

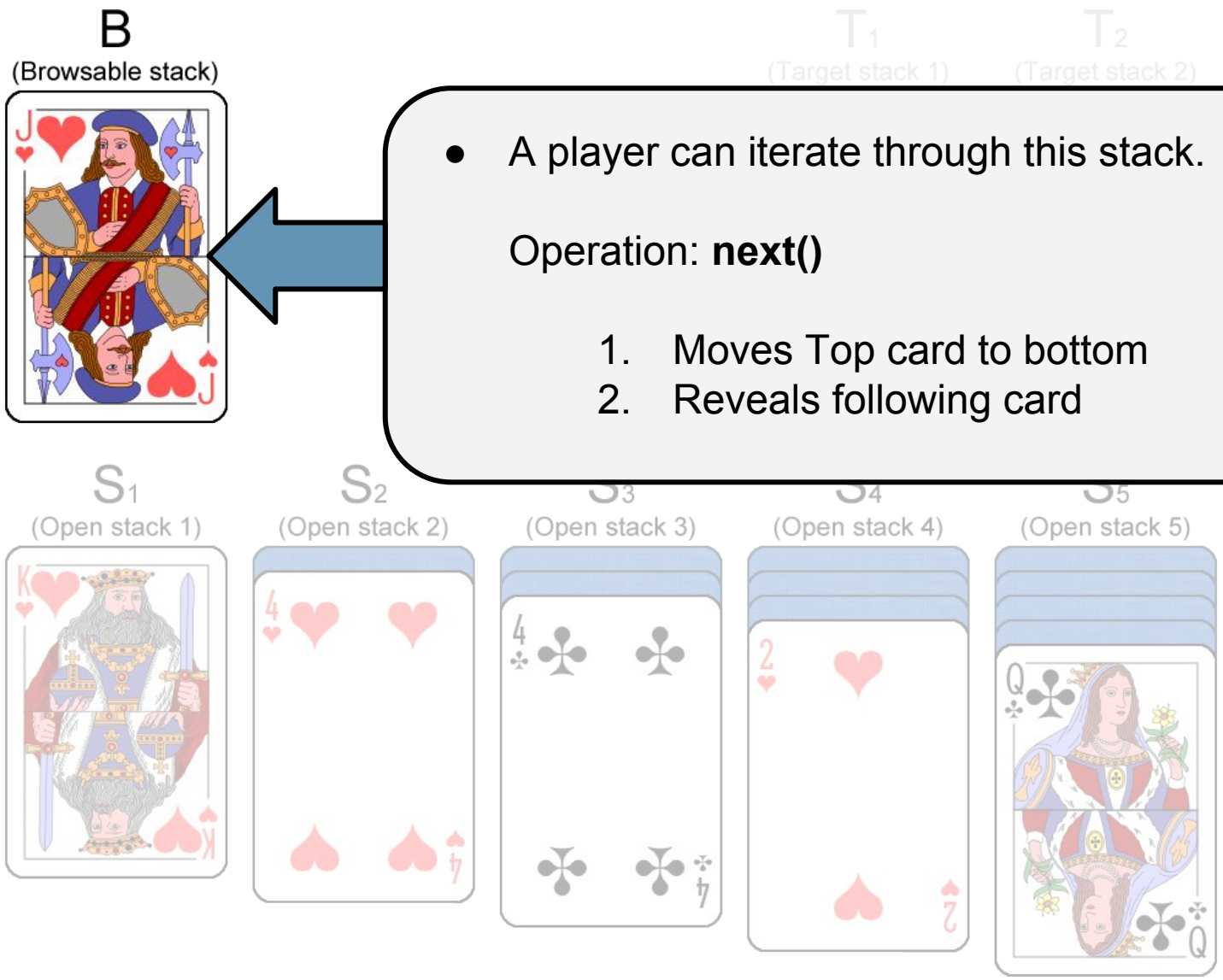


Simplified Solitaire

- **Goal of the game:**
  - Place all cards on T_1 and T_2
  - **Clubs** on T_1
  - **Hearts** on T_2
  - **SORTED!**



Simplified Solitaire

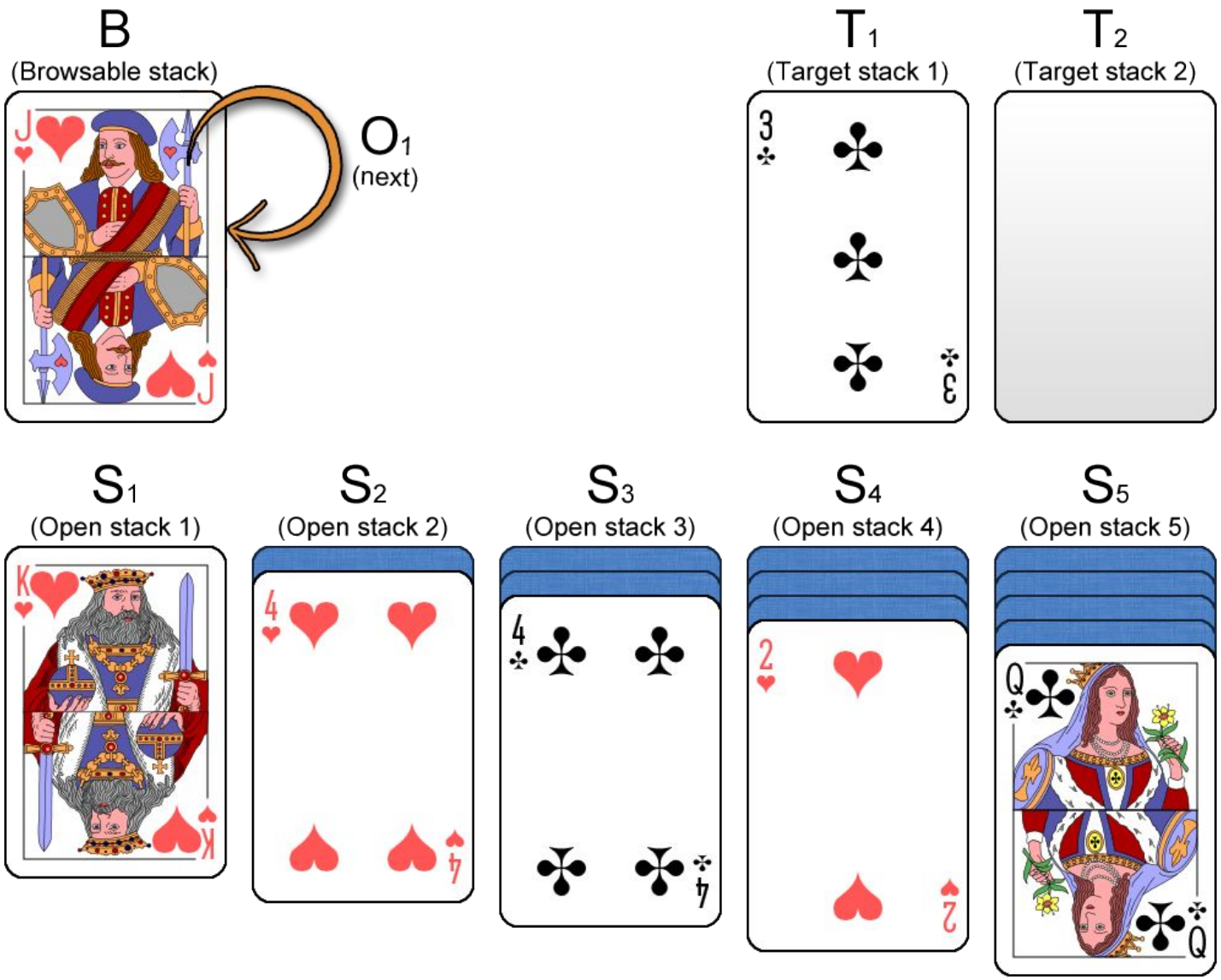


- A player can iterate through this stack.

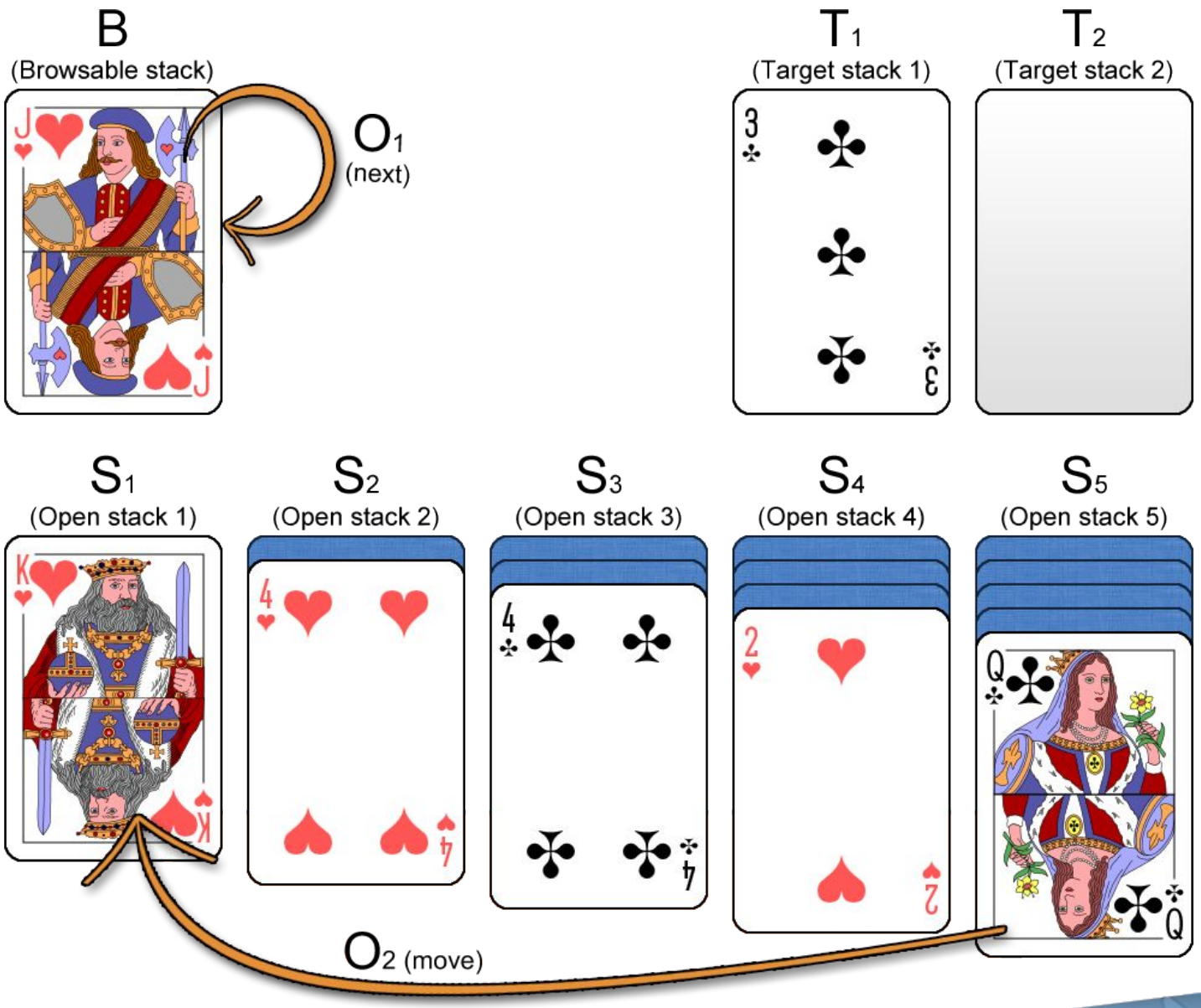
  Operation: **next()**

  1. Moves Top card to bottom
  2. Reveals following card

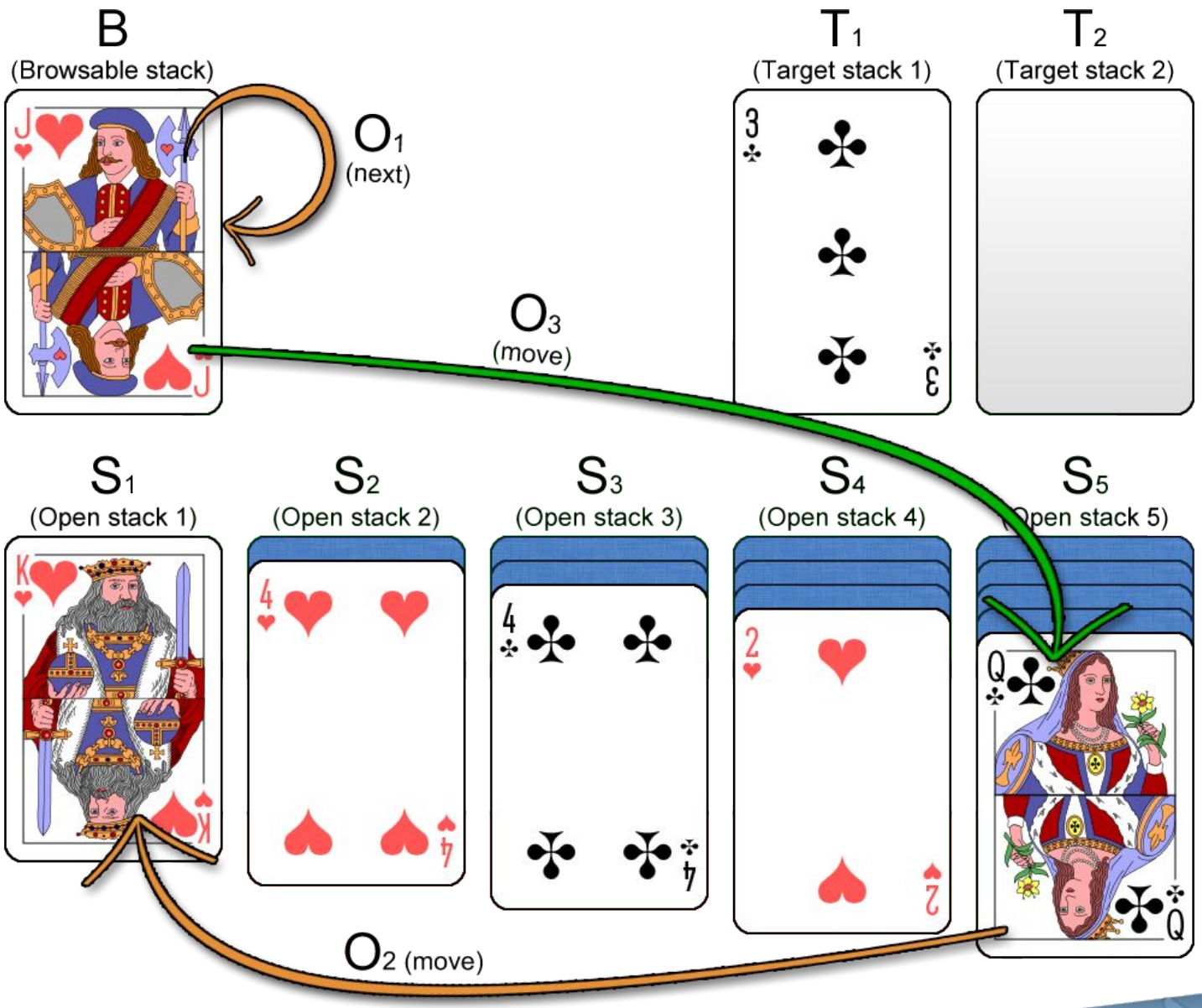
Simplified Solitaire



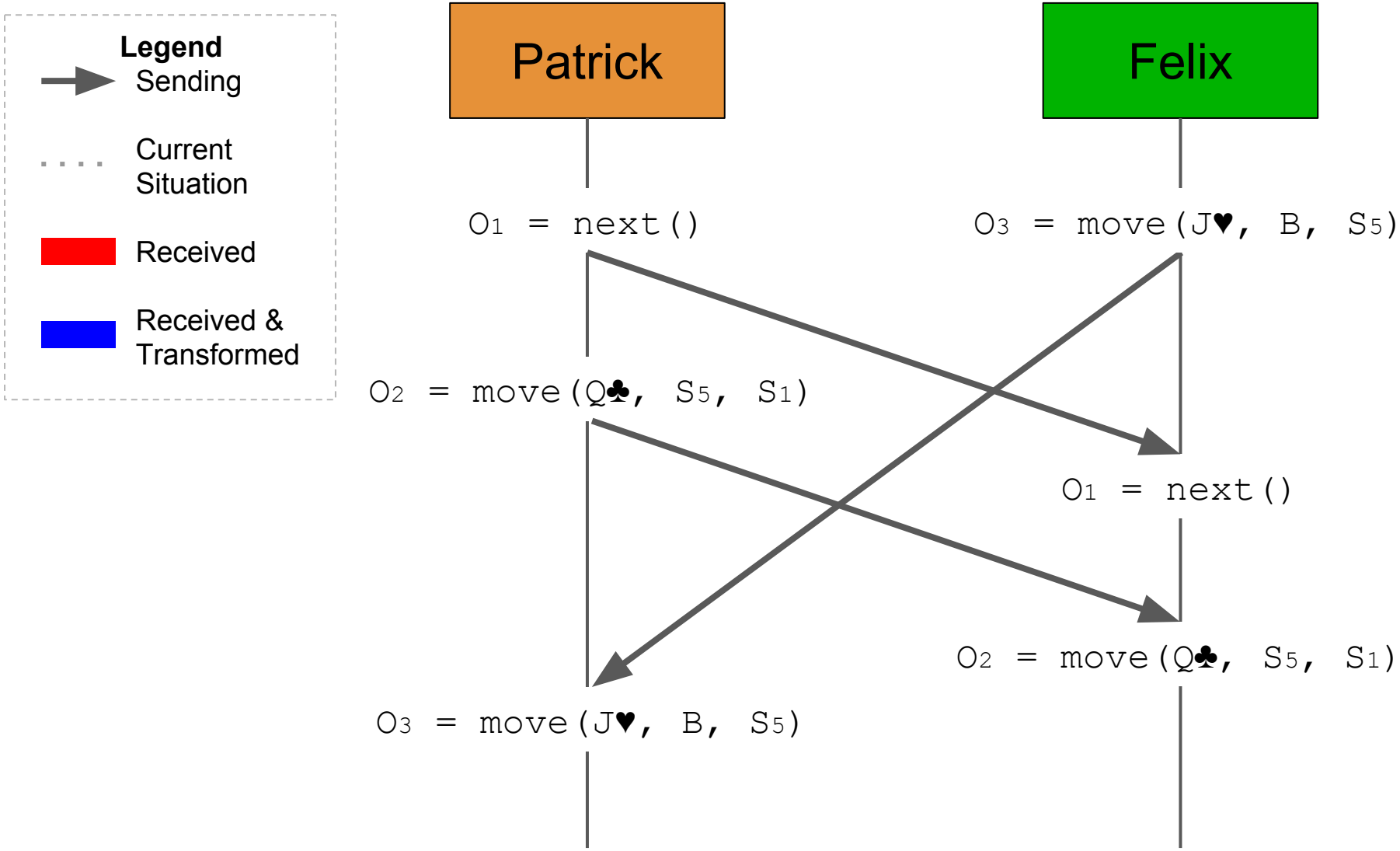
Simplified Solitaire



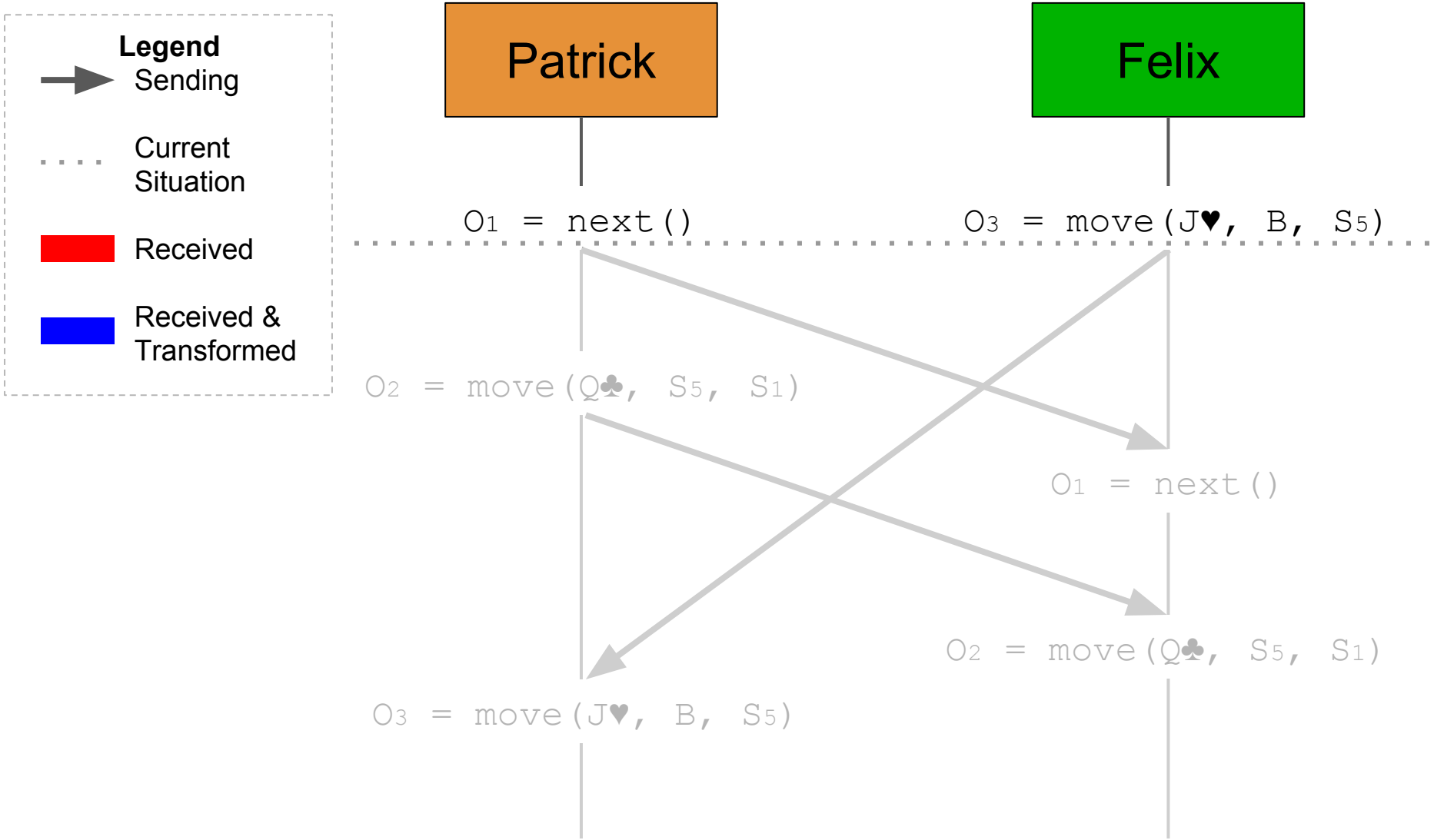
Simplified Solitaire



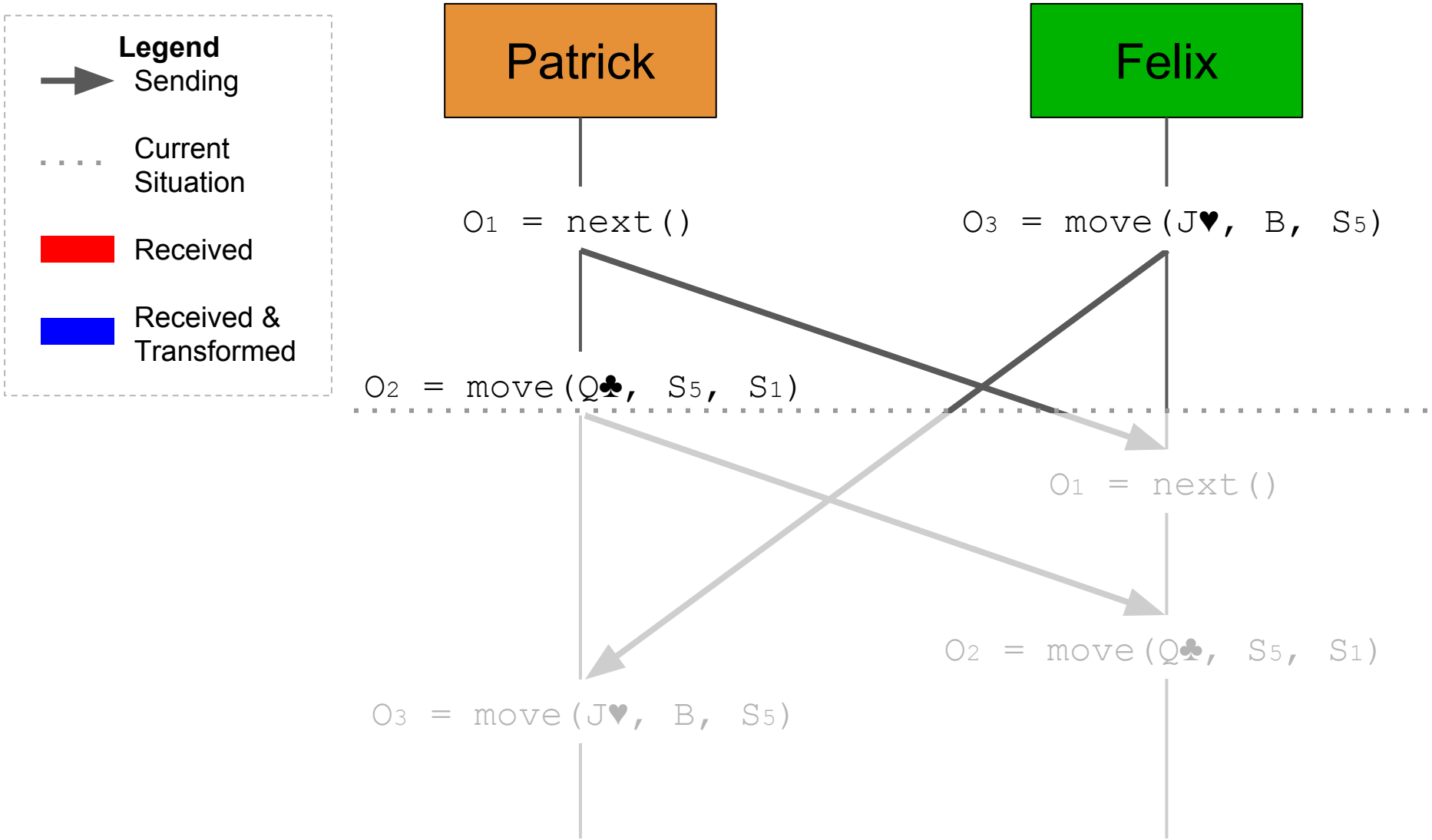
Simplified Solitaire



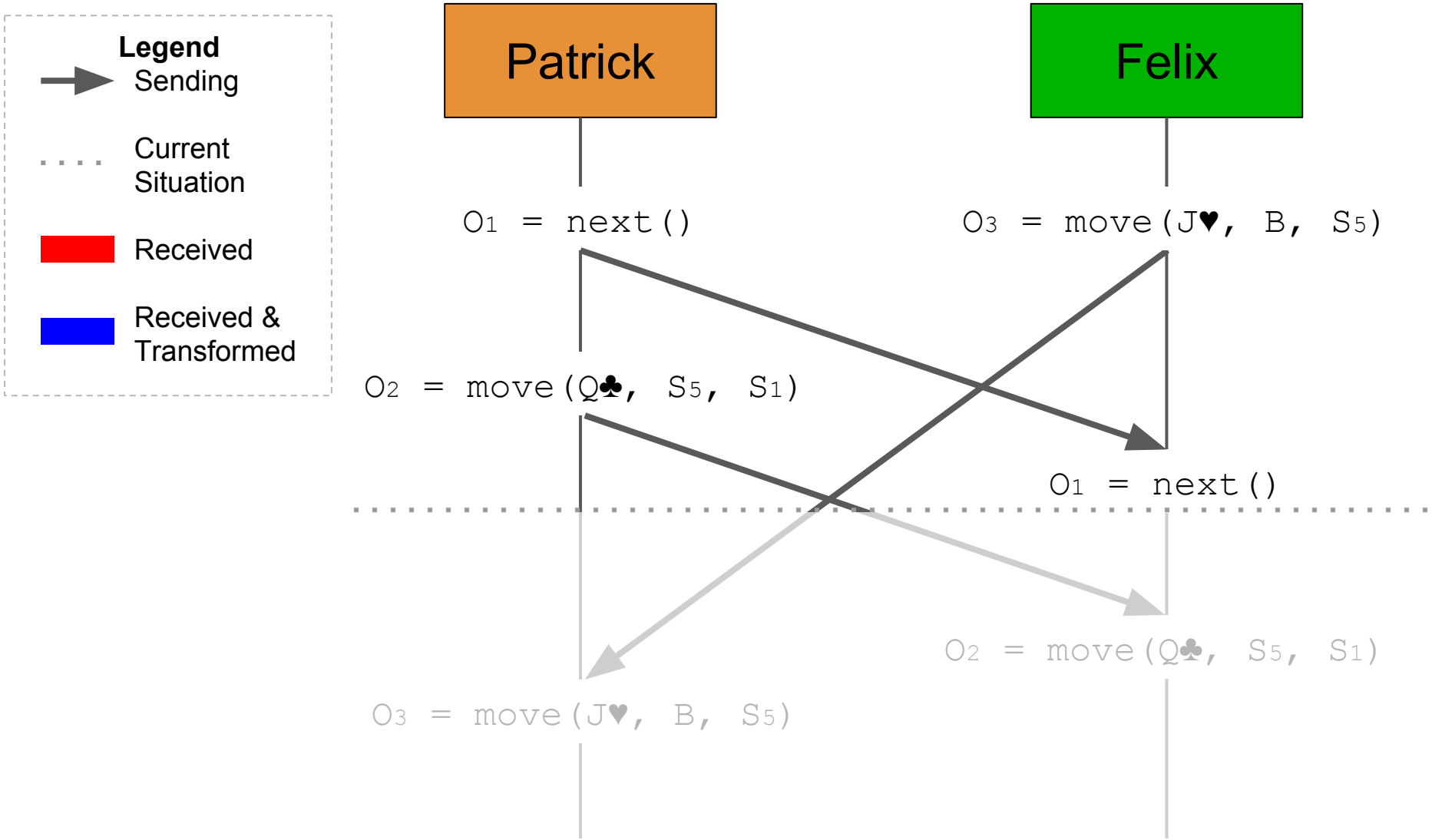
Simplified Solitaire



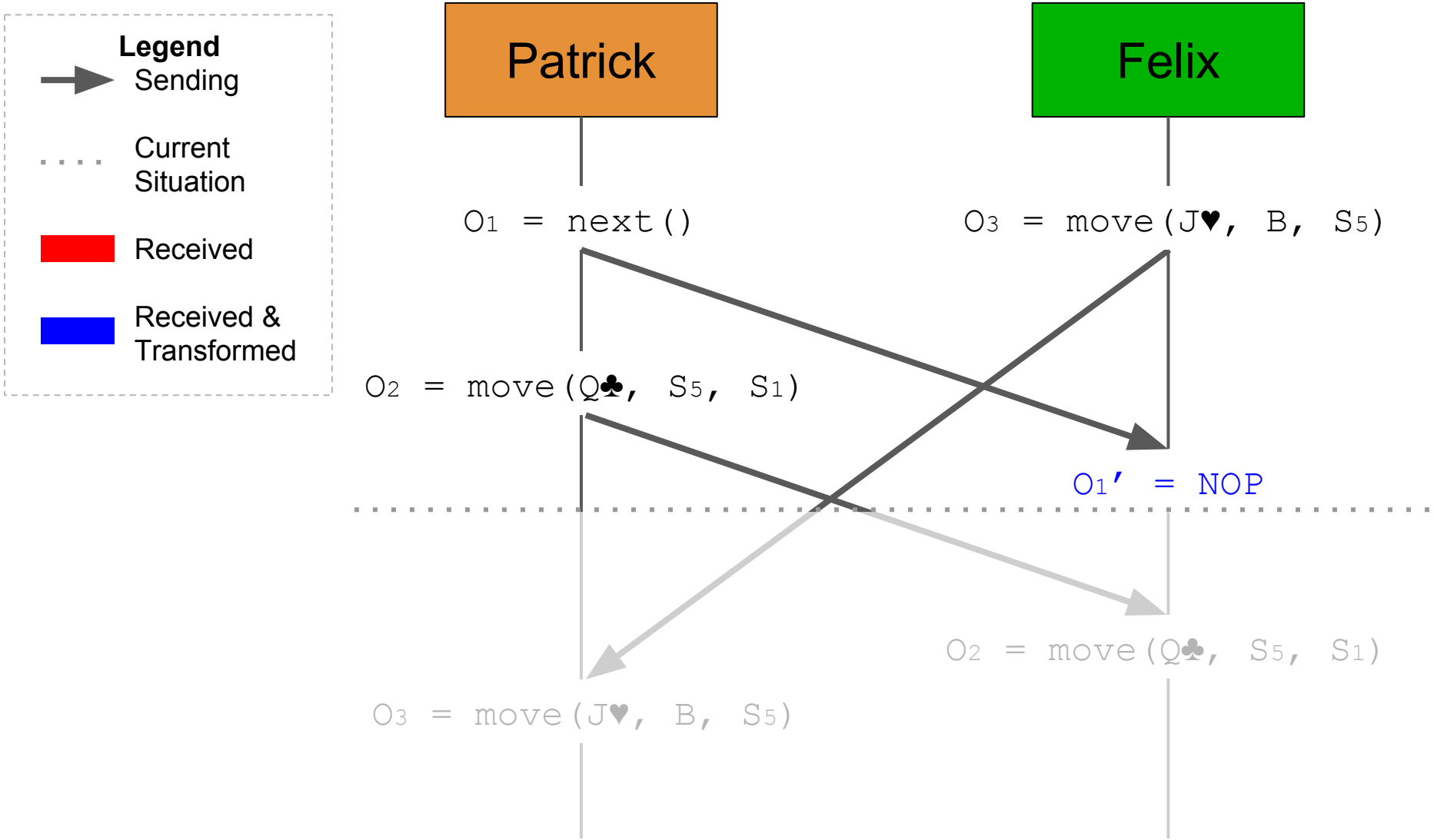
Simplified Solitaire



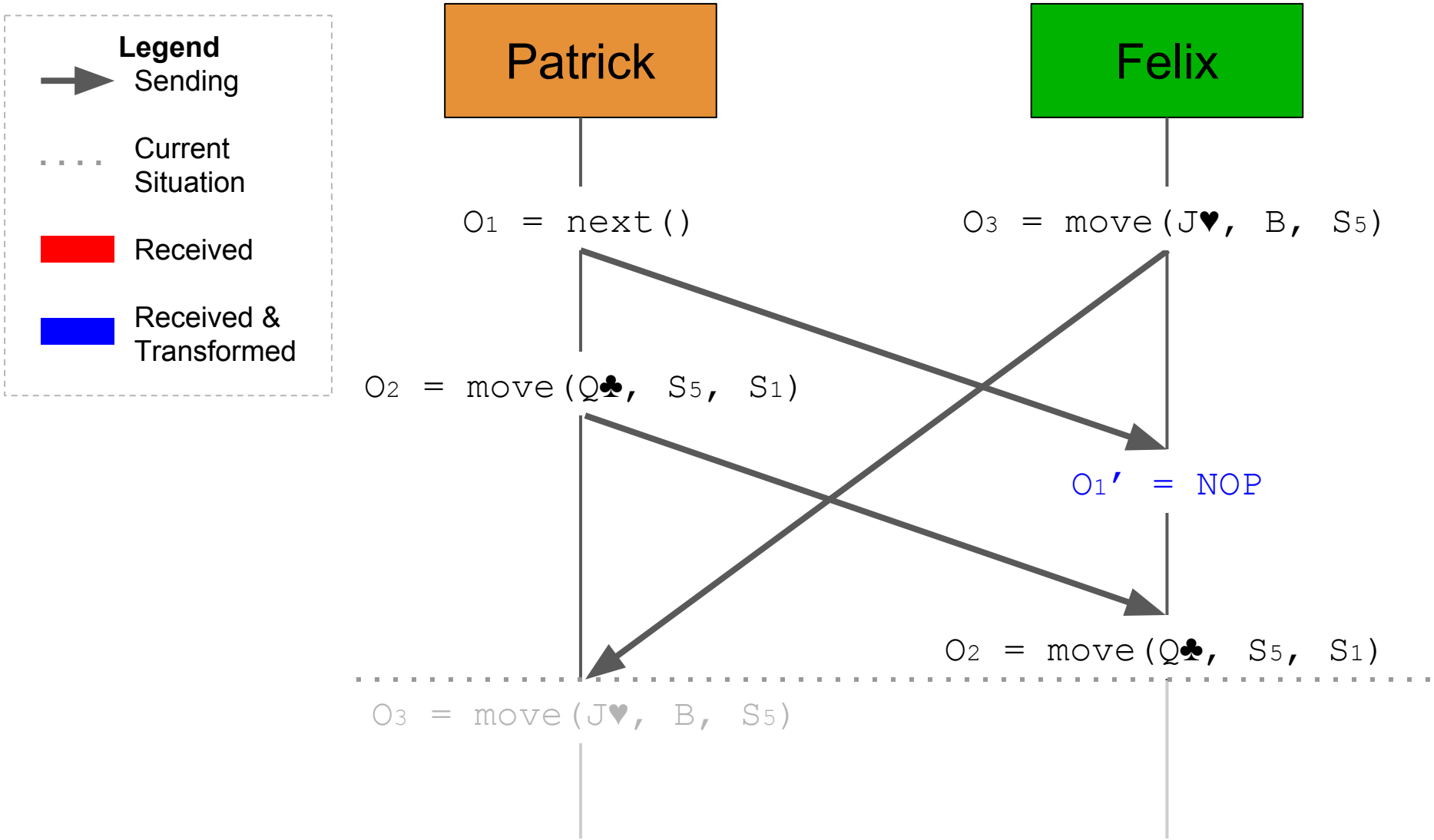
Simplified Solitaire



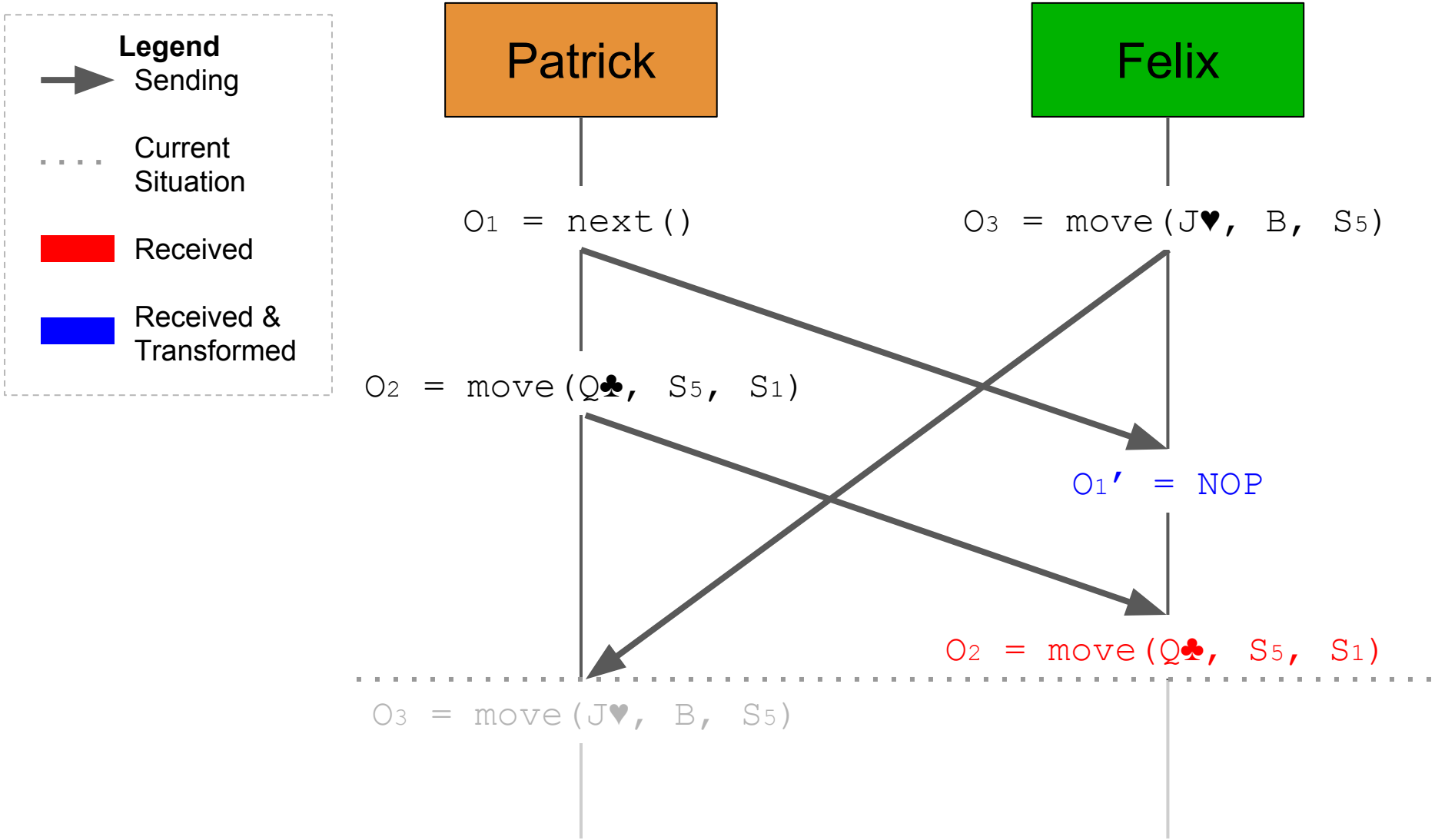
Simplified Solitaire



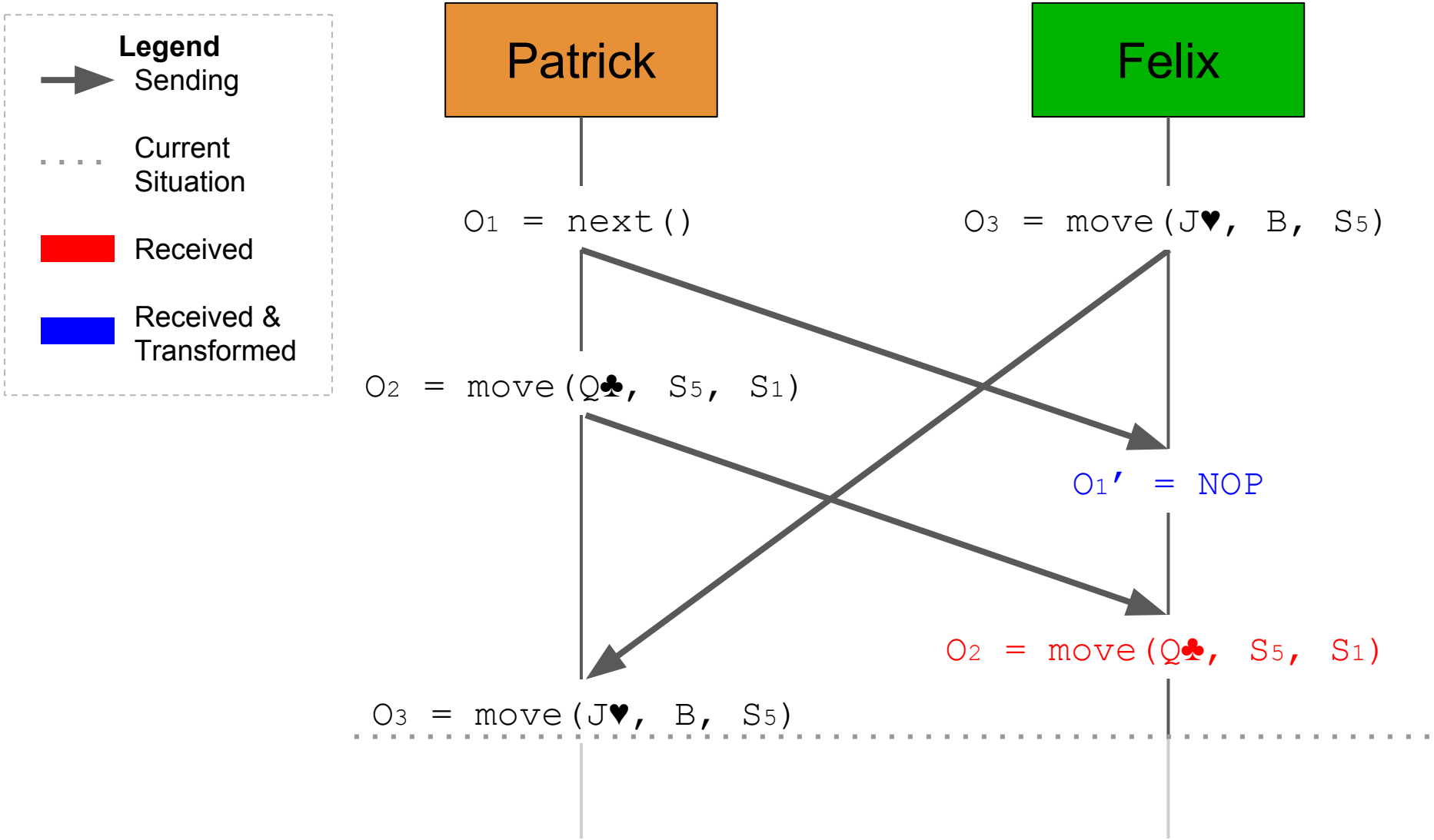
Simplified Solitaire



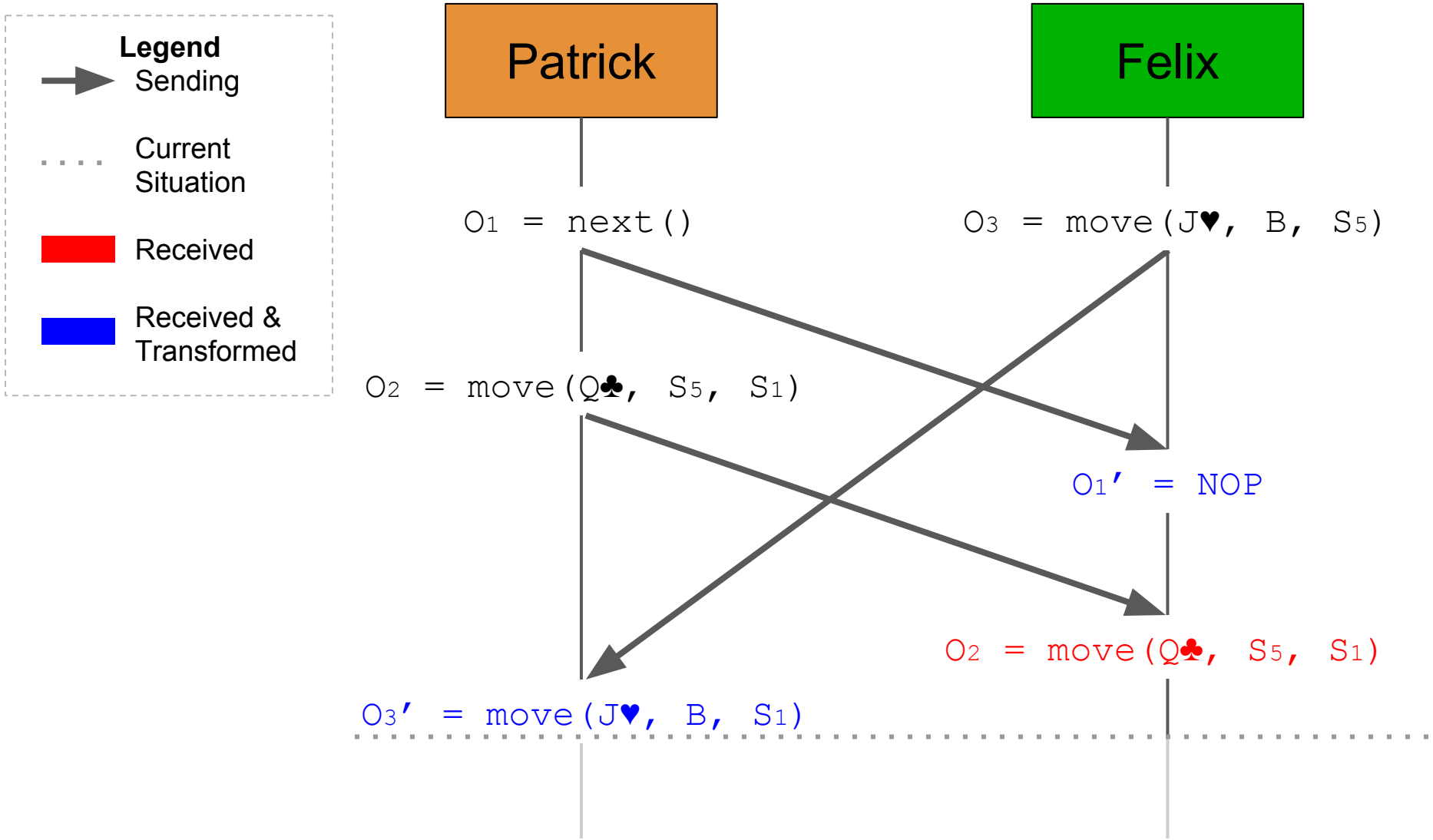
Simplified Solitaire



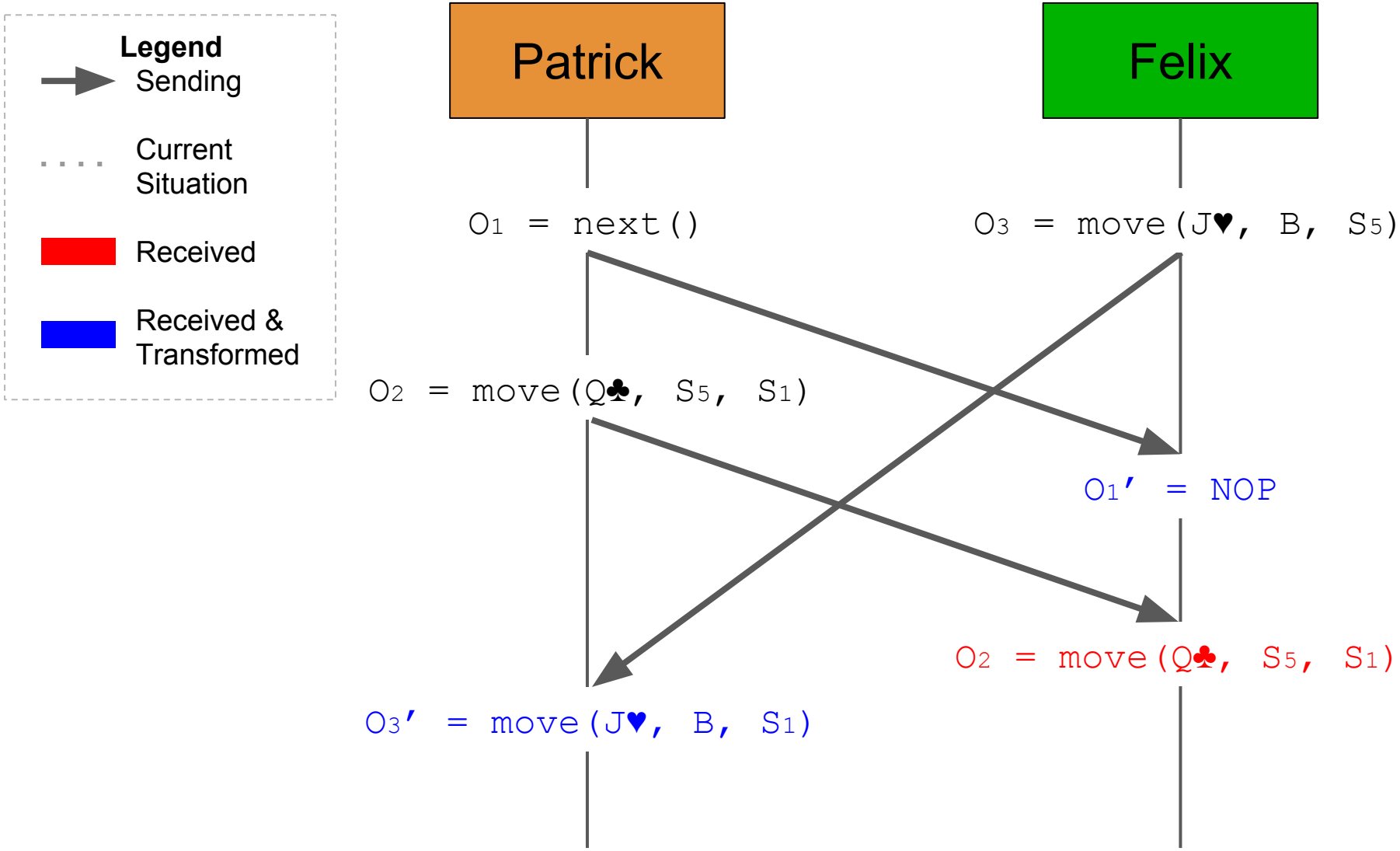
Simplified Solitaire



Simplified Solitaire



Simplified Solitaire



Consistency Models

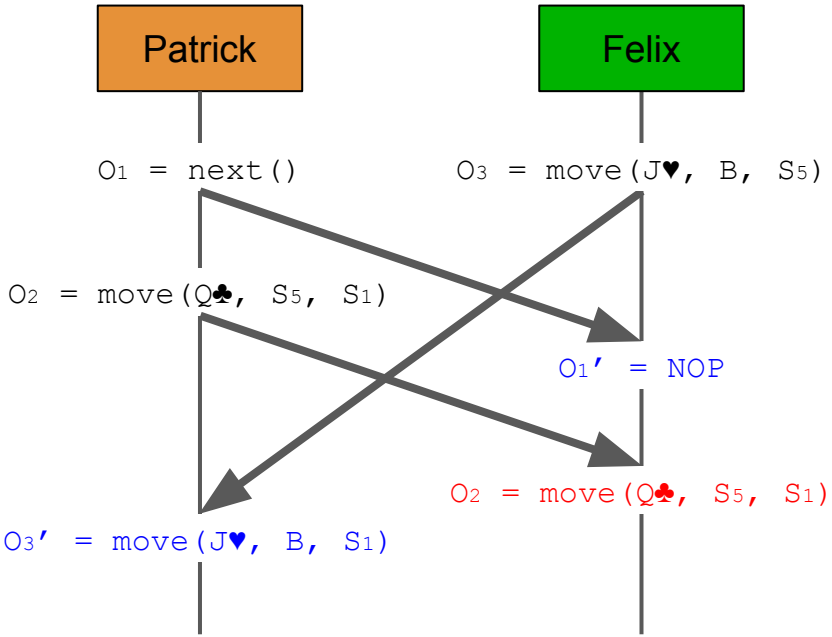
- **CC Model**
  - **C**ausality (Precedence): All dependent operations are executed in the same order at all sites.
  - **C**onvergence: The configuration reached after the execution of a set of operations is the same at all sites.

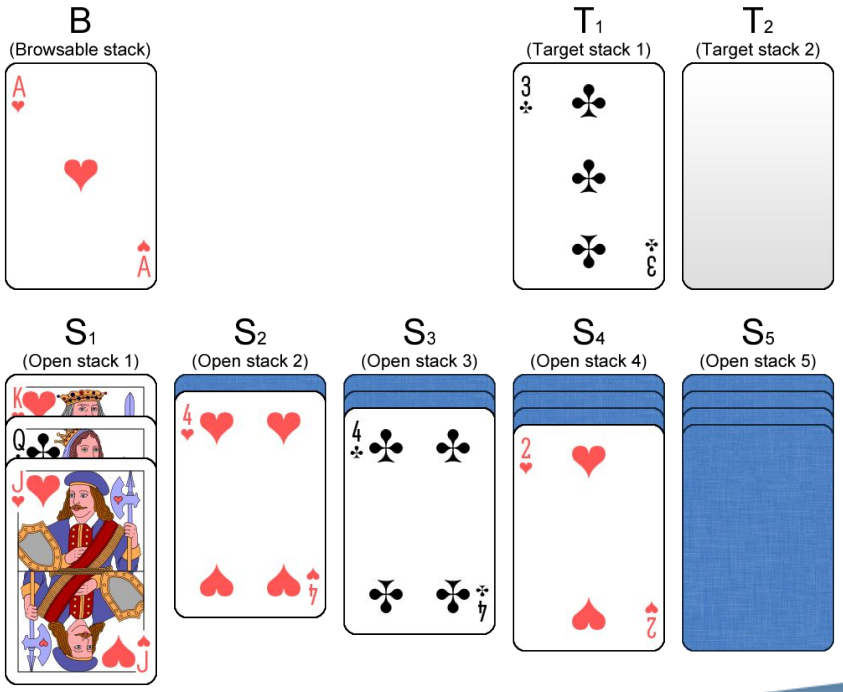
Consistency Models

- **CC Model**

Causality (Precedence): All dependent operations are executed in the same order at all sites.

Convergence: The configuration reached after the execution of a set of operations is the same at all sites.



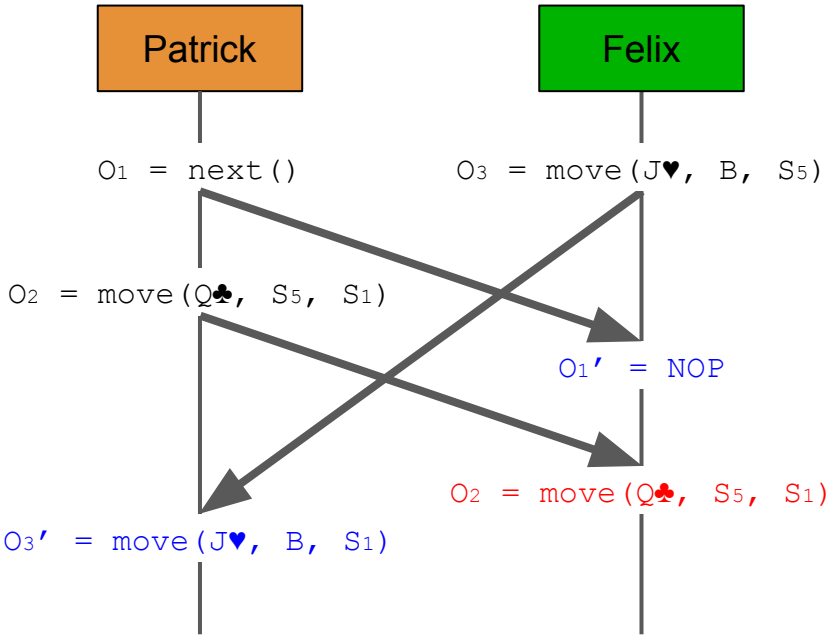


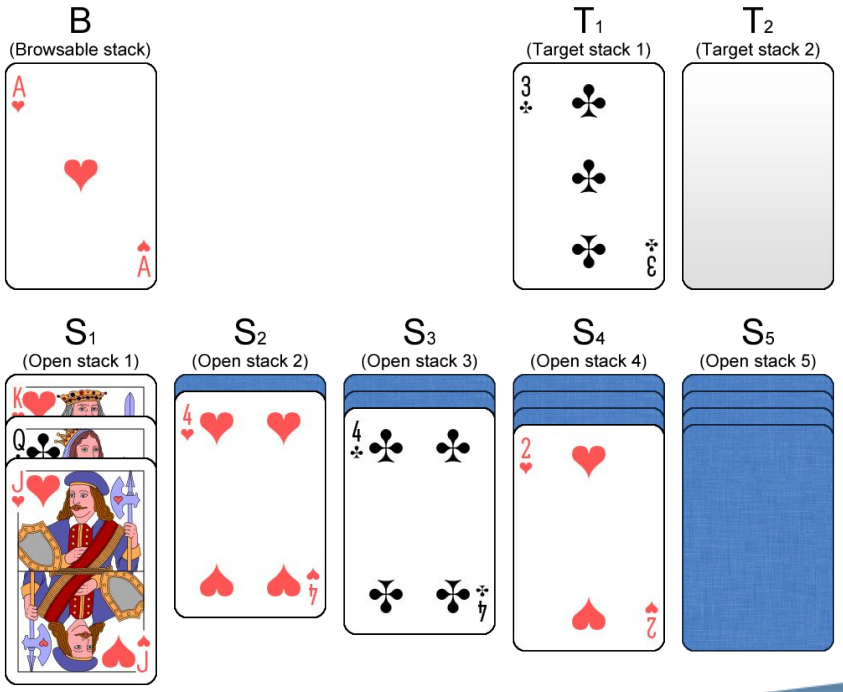
Consistency Models

- **CC Model**

  Causality (Precedence): All dependent operations are executed in the same order at all sites.

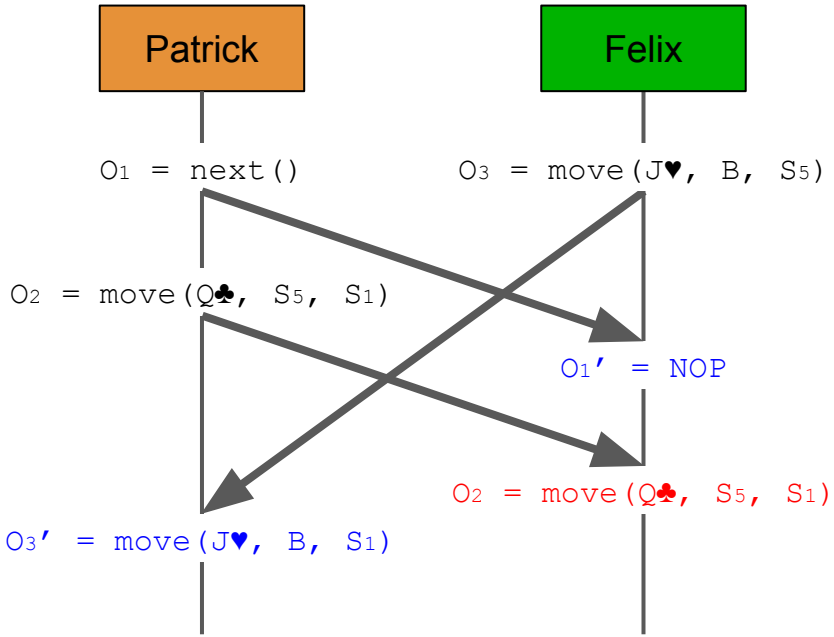
  Convergence: The configuration reached after the execution of a set of operations is the same at all sites.





Consistency Models

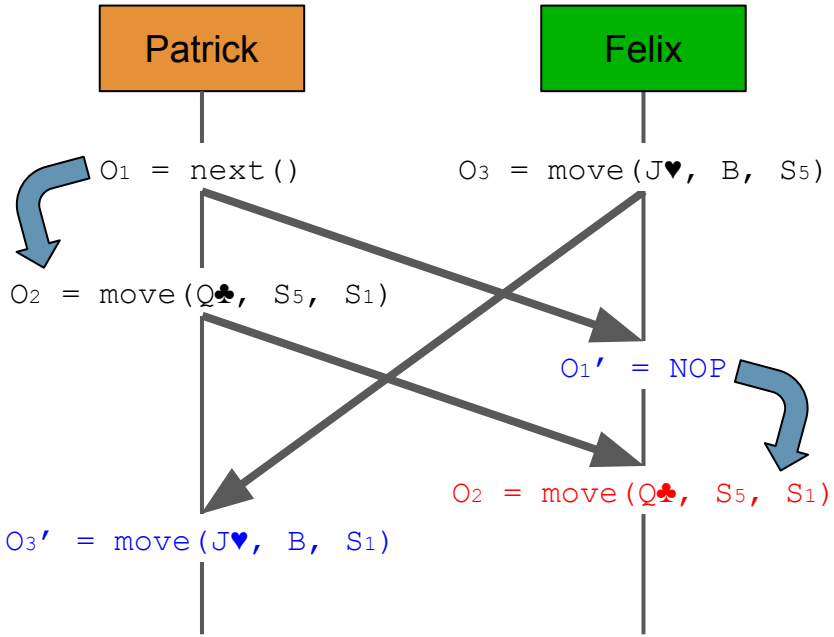
- **CC Model**
  - **C**ausality (Precedence): All dependent operations are executed in the same order at all sites.
  - **C**onvergence: The configuration reached after the execution of a set of operations is the same at all sites.



$$O_1 \rightarrow O_2$$

Consistency Models

- **CC Model**
  - **C**ausality (Precedence): All dependent operations are executed in the same order at all sites.
  - **C**onvergence: The configuration reached after the execution of a set of operations is the same at all sites.



$$O_1 \rightarrow O_2$$

Consistency Models

- **CC Model**
  - **C**ausality (Precedence): All dependent operations are executed in the same order at all sites.
  - **C**onvergence: The configuration reached after the execution of a set of operations is the same at all sites.

All properties satisfied.

Consistency Models

- **CC Model**
  - **C**ausality (Precedence): All dependent operations are executed in the same order at all sites.
  - **C**onvergence: The configuration reached after the execution of a set of operations is the same at all sites.

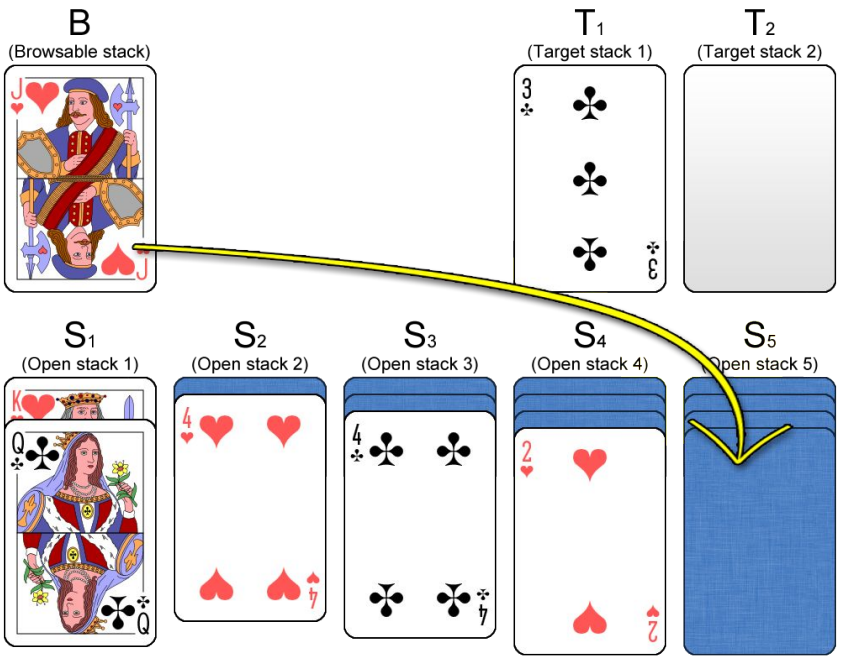
All properties satisfied.

BUT: What happens if...

Consistency Models

- **CC Model**
  - **C**ausality (Precedence): All dependent operations are executed in the same order at all sites.
  - **C**onvergence: The configuration reached after the execution of a set of operations is the same at all sites.

- Patrick moved the **Queen of Clubs** to **S_1**
- Felix moves the **Jack of Hearts** to **S_5**

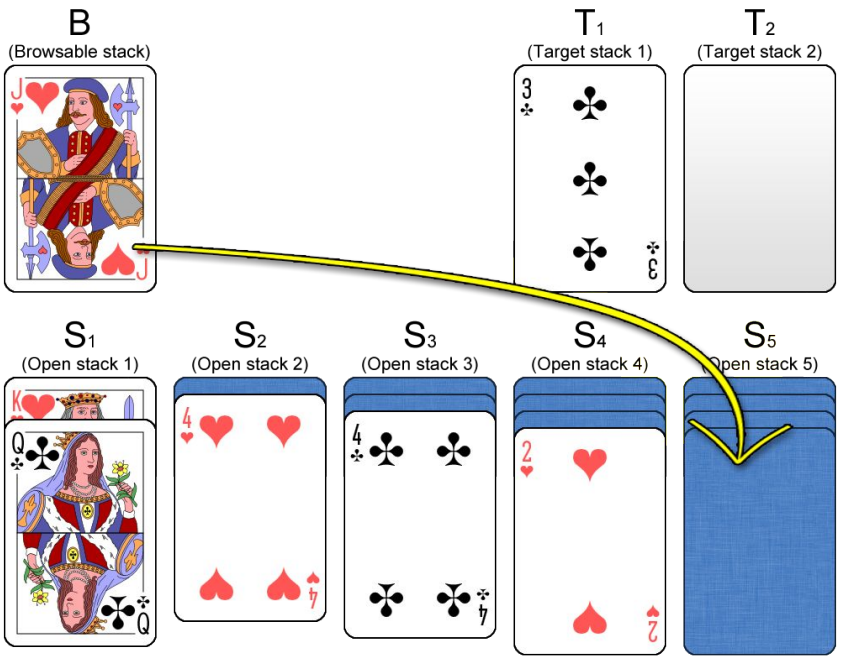


Consistency Models

- **CC Model**
  - **C**ausality (Precedence): All dependent operations are executed in the same order at all sites.
  - **C**onvergence: The configuration reached after the execution of a set of operations is the same at all sites.

- Patrick moved the **Queen of Clubs** to **S_1**
- Felix moves the **Jack of Hearts** to **S_5**

Violates game rules

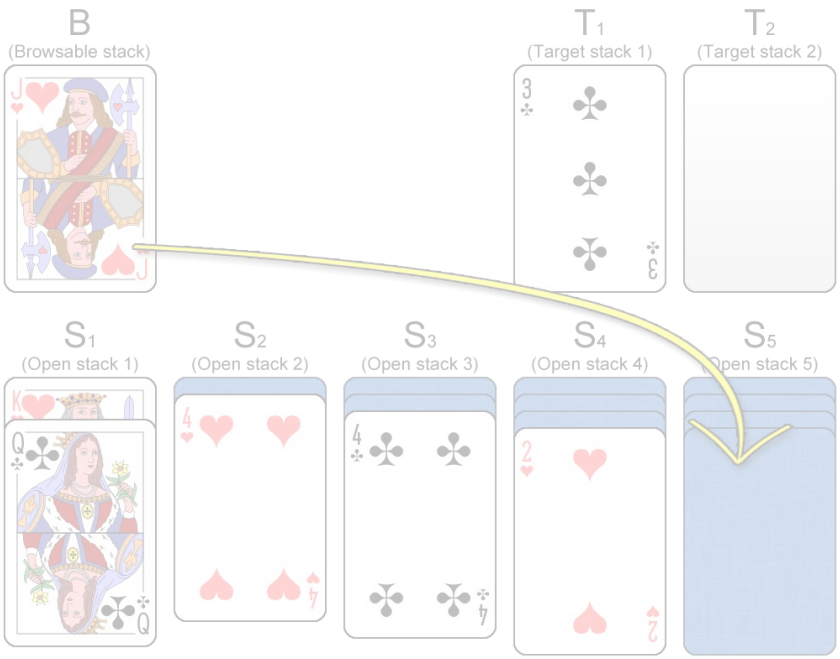


Consistency Models

- **CCI Model** (Extension of CC Model)

  Intention Preservation: The Intention of any executed operation is always the same

- Patrick moved the **Queen of Clubs** to **S_1**
- Felix moves the **Jack of Hearts** to **S_5**

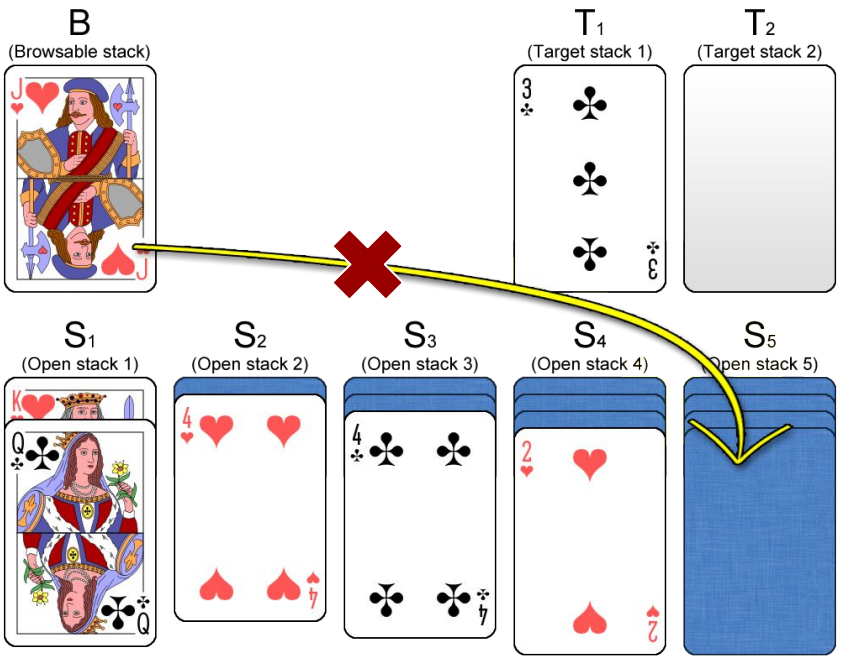


Consistency Models

- **CCI Model** (Extension of CC Model)

  Intention Preservation: The Intention of any executed operation is always the same

- Patrick moved the **Queen of Clubs** to **S_1**
- Felix moves the **Jack of Hearts** to **S_5**



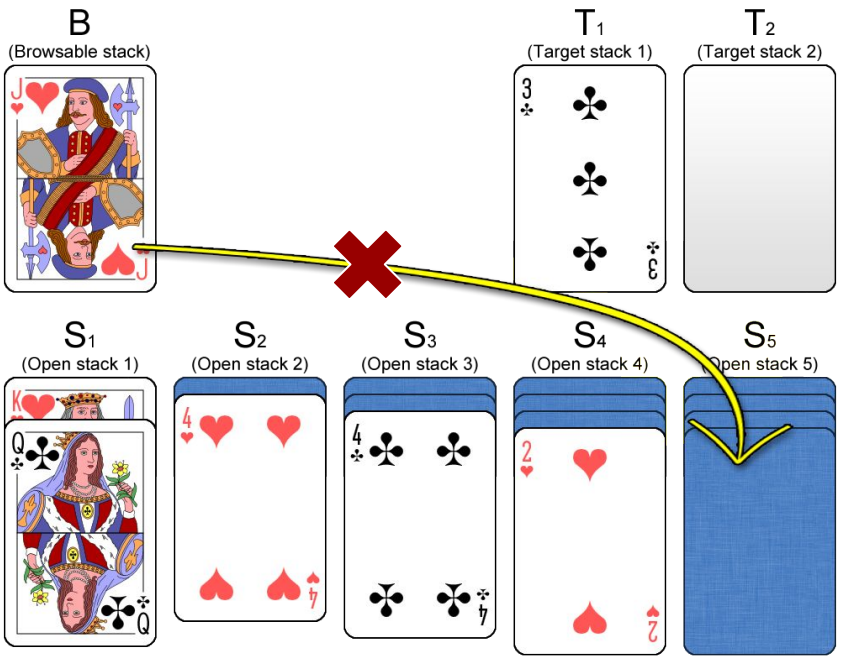
Consistency Models

- **CCI Model** (Extension of CC Model)

  Intention Preservation: The Intention of any executed operation is always the same

- Patrick moved the **Queen of Clubs** to **S_1**
- Felix moves the **Jack of Hearts** to **S_5**

Transformation

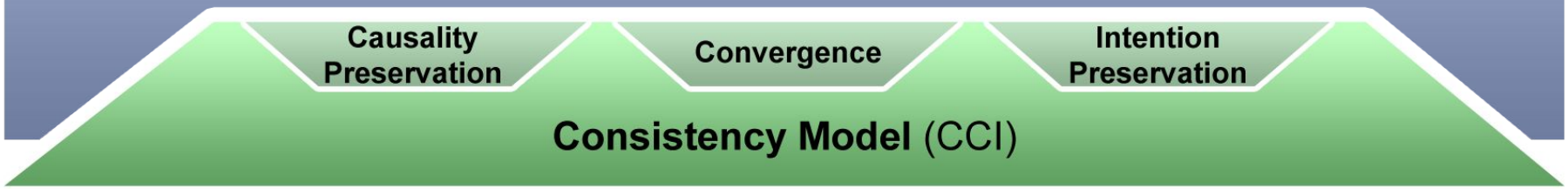


OT Algorithm

OT Algorithm

OT Algorithm

OT Algorithm



OT Algorithm

OT Algorithm

Causality Preservation

Convergence

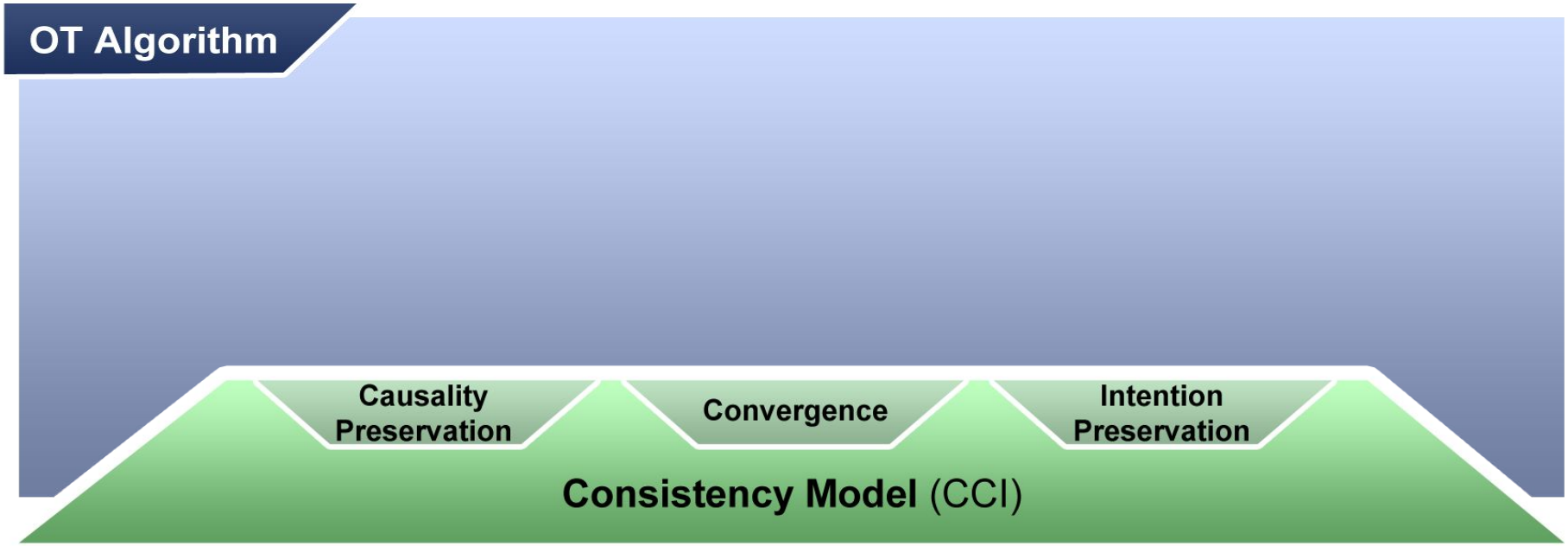
Intention Preservation

Consistency Model (CCI)

Properties

OT Algorithm

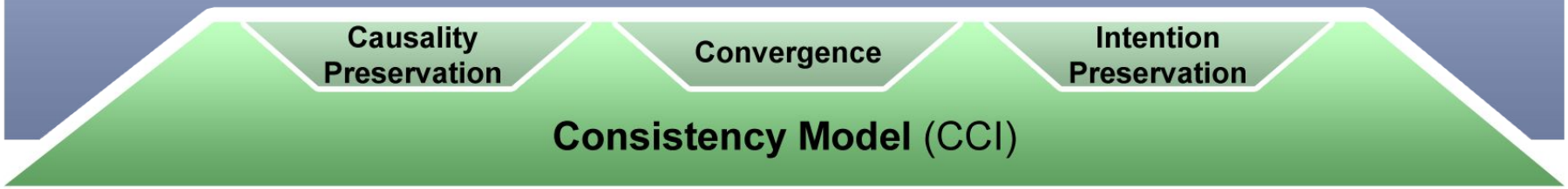
OT Algorithm



Properties → **Immediate results**

OT Algorithm

OT Algorithm

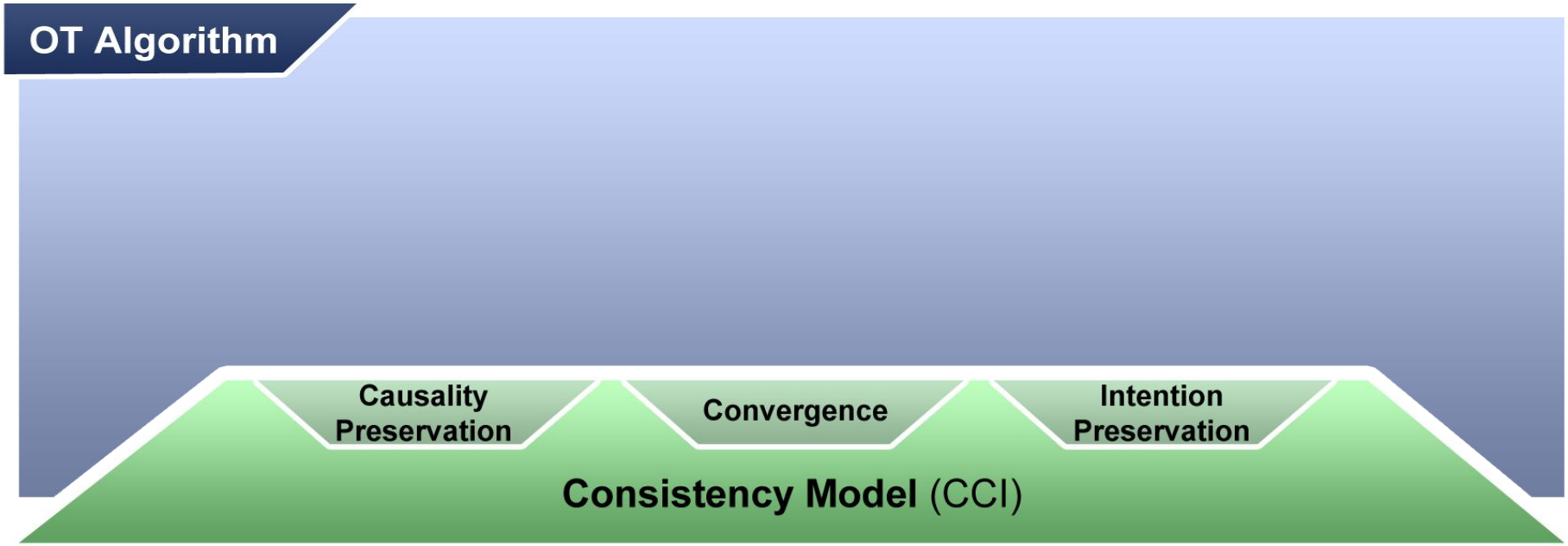


Properties

- **Immediate results**
- **No locking**

OT Algorithm

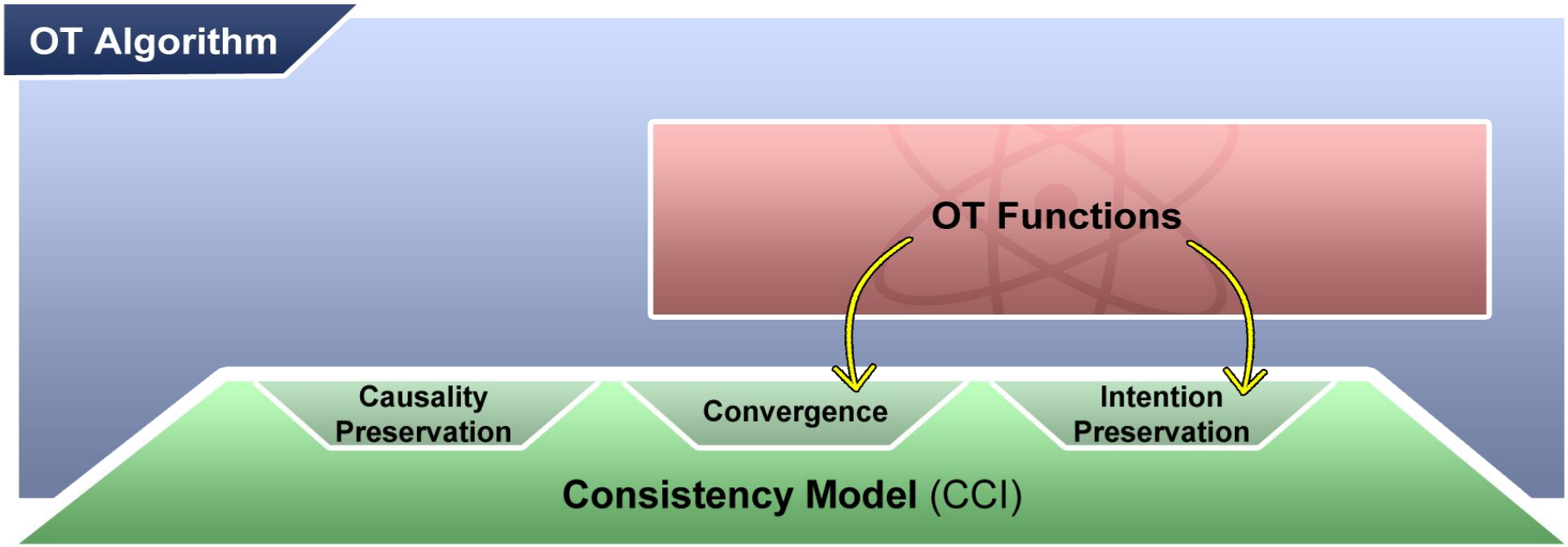
OT Algorithm



Properties

- **Immediate results**
- **No locking**
- **Fully distributive**

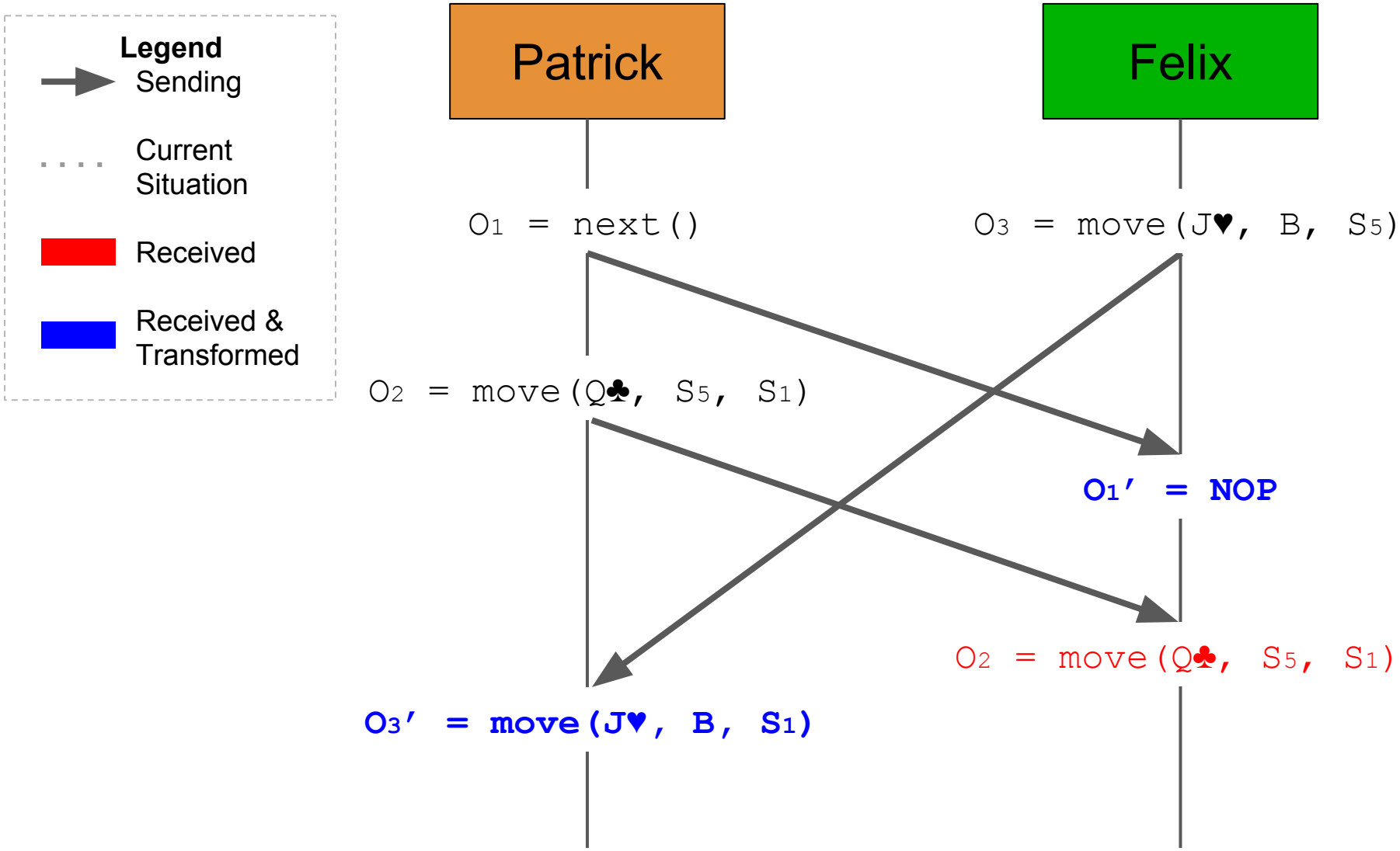
OT Algorithm



Properties

- Immediate results
- No locking
- Fully distributive

Simplified Solitaire



OT Functions: Example 1

`next()` → Transform against → `move(J♥, B, S5)`

OT Functions: Example 1

`next()` → Transform against → `move(J♥, B, S5)`

- Transformation Matrix:

	move	**next**
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$T_{next,\ move}$	$T_{next,\ next}$

OT Functions: Example 1

`next()` → Transform against → `move(J♥, B, S5)`

- Transformation Matrix:

	move	**next**
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$\boxed{T_{next,\ move}}$	$T_{next,\ next}$

OT Functions: Example 1

`next()` → Transform against → `move(J♥, B, S5)`

- Transformation Matrix:

	move	**next**
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$\boxed{T_{next,\ move}}$	$T_{next,\ next}$

- Function call:

$\mathbf{T}_{\mathtt{next,move}}$`(next(), move(J♥, B, S5))`

OT Functions: Example 1

`next()` → Transform against → `move(J♥, B, S5)`

- Transformation Matrix:

	move	next
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$\boxed{T_{next,\ move}}$	$T_{next,\ next}$

- Function call:

```
Tnext,move(next(), move(J♥, B, S5))
```

- Function definition:

```
Tnext_move(next(), move(card, from, to)):
    if(from = B)
        return NOP
```

OT Functions: Example 1

`next()` → Transform against → `move(J♥, B, S5)`

- Transformation Matrix:

	move	next
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$\boxed{T_{next,\ move}}$	$T_{next,\ next}$

- Function call:

```
Tnext,move(next(), move(J♥, B, S5))
```

- Function definition:

```
Tnext_move(next(), move(card, from, to)):
    if(from = B)
        return NOP
```

OT Functions: Example 1

`next()` → Transform against → `move(J♥, B, S5)`

- Transformation Matrix:

	move	next
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$\boxed{T_{next,\ move}}$	$T_{next,\ next}$

- Function call:

```
Tnext,move(next(), move(J♥, B, S5))
```

- Function definition:

```
Tnext_move(next(), move(card, from, to)):
    if(from = B)
        return NOP
```

OT Functions: Example 2

`move(J♥, B, S5)` → Transform against → `move(Q♣, S5, S1)`

- Transformation Matrix:

	move	**next**
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$T_{next,\ move}$	$T_{next,\ next}$

- Function call:

- Function definition:

OT Functions: Example 2

`move(J♥, B, S5)` → Transform against → `move(Q♣, S5, S1)`

- Transformation Matrix:

	move	next
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$T_{next,\ move}$	$T_{next,\ next}$

- Function call:

$T_{move,move}$(`move(J♥, B, S5)`, `move(Q♣, S5, S1)`)

- Function definition:

```
Tmove,move(move(card1, from1, to1), move(card2, from2, to2)):
    if(to1 = from2)
        return move(card1, from1, to2)
```

OT Functions: Example 2

`move(J♥, B, S5)` → Transform against → `move(Q♣, S5, S1)`

- Transformation Matrix:

	move	next
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$T_{next,\ move}$	$T_{next,\ next}$

- Function call:

$T_{move,move}$(move(J♥, B, S5), move(Q♣, S5, S1))

- Function definition:

```
Tmove,move(move(card1, from1, to1), move(card2, from2, to2)):
    if(to1 = from2)
        return move(card1, from1, to2)
```

OT Functions: Example 2

`move(J♥, B, S5)` → Transform against → `move(Q♣, S5, S1)`

- Transformation Matrix:

	move	next
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$T_{next,\ move}$	$T_{next,\ next}$

- Function call:

$T_{move,move}$(move(J♥, B, S5), move(Q♣, S5, S1))

- Function definition:

```
Tmove,move(move(card1, from1, to1), move(card2, from2, to2)):
    if(to1 = from2)
        return move(card1, from1, to2)
        //      move( J♥   ,       ,     )
```

OT Functions: Example 2

`move(J♥, B, S5)` → Transform against → `move(Q♣, S5, S1)`

- Transformation Matrix:

	move	next
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$T_{next,\ move}$	$T_{next,\ next}$

- Function call:

$T_{move,move}$(move(J♥, B, S5), move(Q♣, S5, S1))

- Function definition:

```
Tmove,move(move(card1, from1, to1), move(card2, from2, to2)):
    if(to1 = from2)
        return move(card1, from1, to2)
        //      move( J♥   , B     ,     )
```

OT Functions: Example 2

`move(J♥, B, S5)` → Transform against → `move(Q♣, S5, S1)`

- Transformation Matrix:

	move	next
move	$T_{move,\ move}$	$T_{move,\ next}$
next	$T_{next,\ move}$	$T_{next,\ next}$

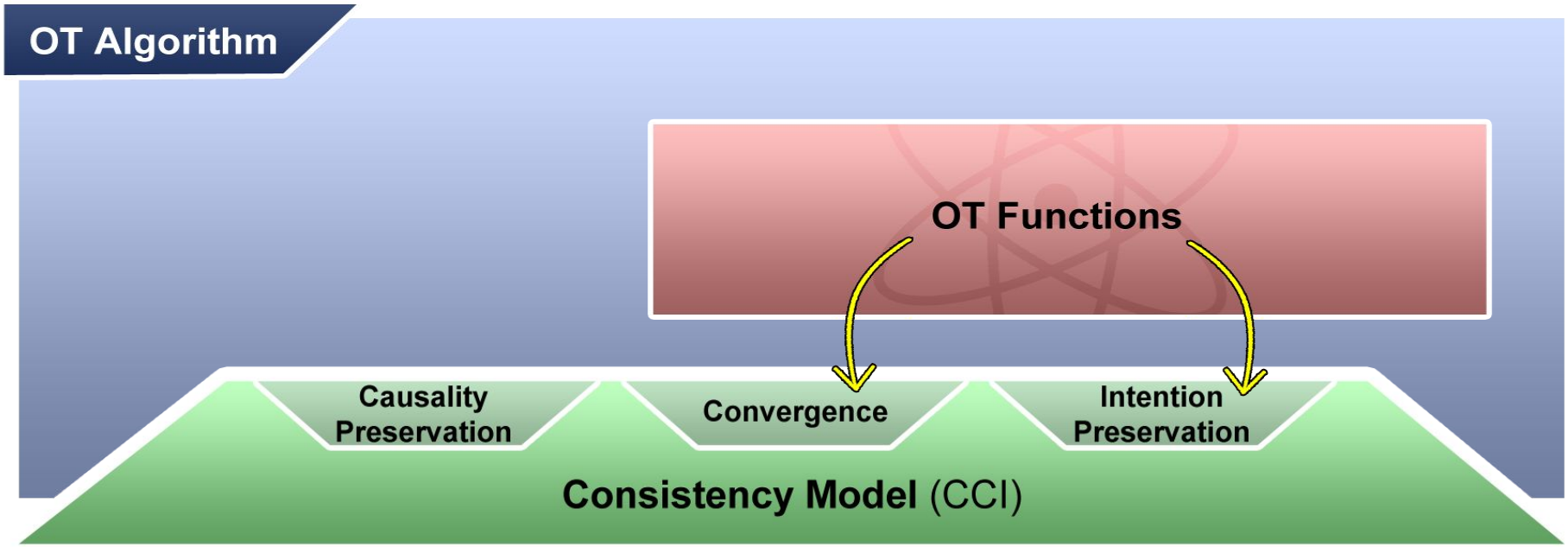
- Function call:

$T_{move,move}$(move(J♥, B, S5), move(Q♣, S5, S1))

- Function definition:

```
Tmove,move(move(card1, from1, to1), move(card2, from2, to2)):
    if(to1 = from2)
        return move(card1, from1, to2)
        //      move( J♥  , B     , S1 )
```

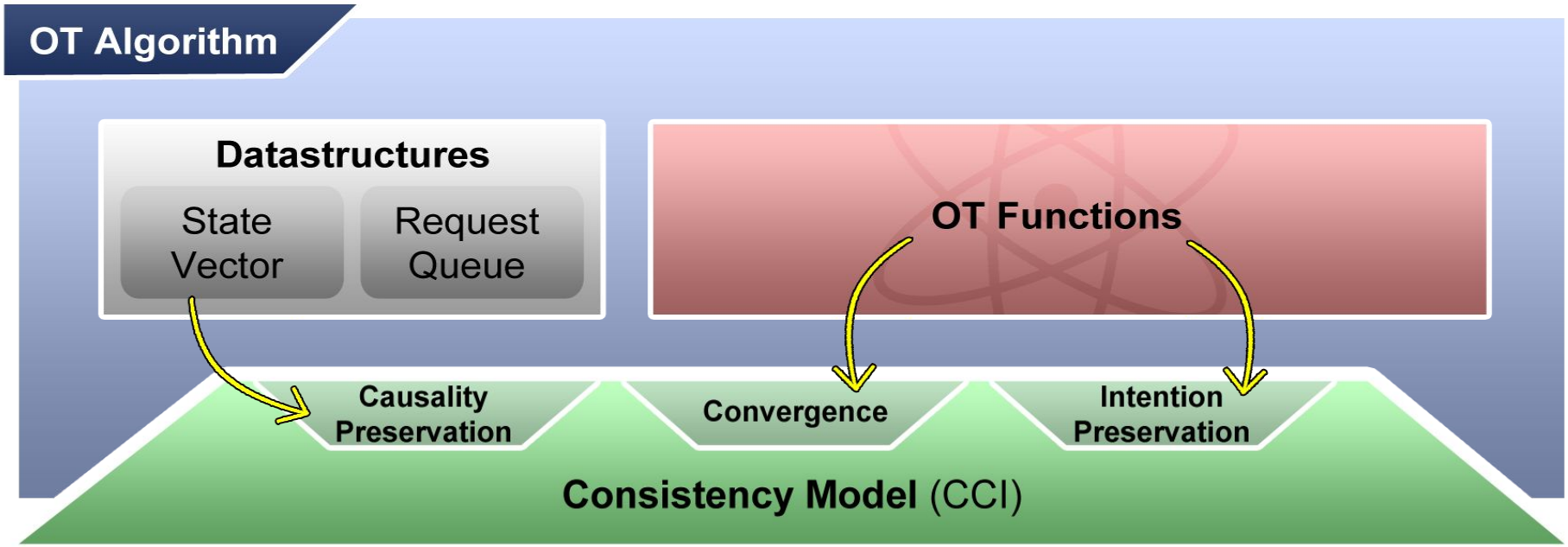
OT Algorithm



Properties

- Immediate results
- No locking
- Fully distributive

OT Algorithm



Properties

- **Immediate results**
- **No locking**
- **Fully distributive**

Collaborative Games

Collaborative games

One goal solved by **many**

VS

Cooperative games

Many goals each solved by **one**

Collaborative Games

Collaborative games *VS* **Cooperative** games

One goal solved by **many**

Many goals each solved by **one**

Are such games available at all?

Collaborative Games

Turn-Based	Single-Object	Multi-Object
No		
Yes		

Collaborative Games

Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	
Yes		

Collaborative Games

Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	
Yes		

Collaborative Games

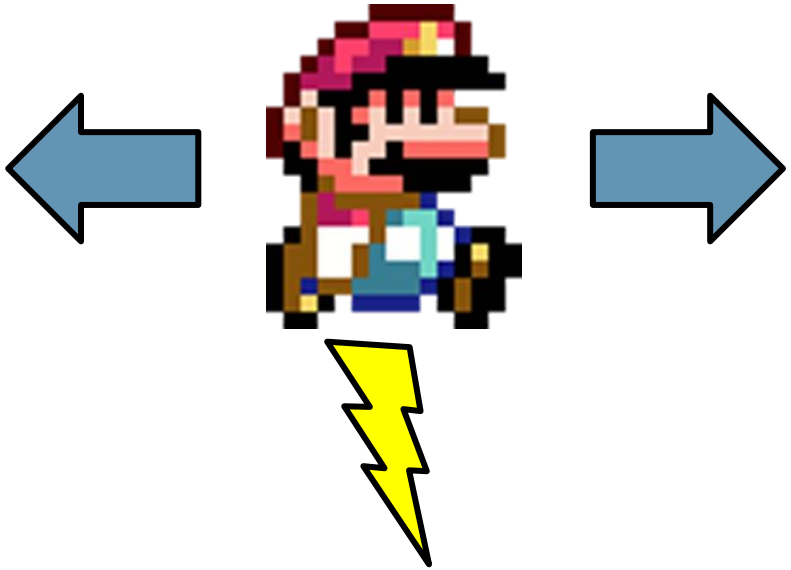
Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	
Yes		

Collaborative Games

Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	
Yes		

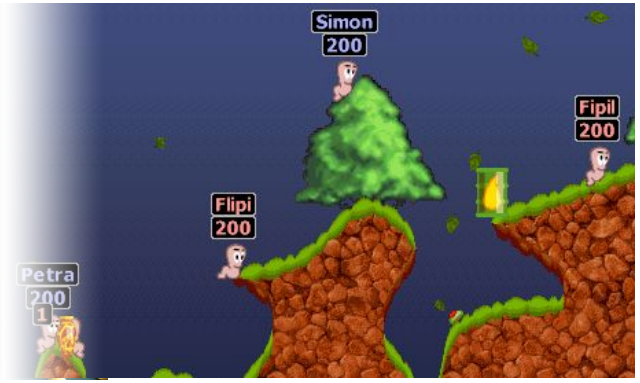
Collaborative Games

Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	
Yes		



Collaborative Games

Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	
Yes	- Chess, Monopoly, Worms, ...	



Collaborative Games

Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	
Yes	- Chess, Monopoly, Worms, ...	

Only 1 Operation at a time

Collaborative Games

Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	
Yes	- Chess, Monopoly, Worms, ...	0 Scrabble, Rummy, ...

Collaborative Games

Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	
Yes	- Chess, Monopoly, Worms, ...	0 Scrabble, Rummy, ...

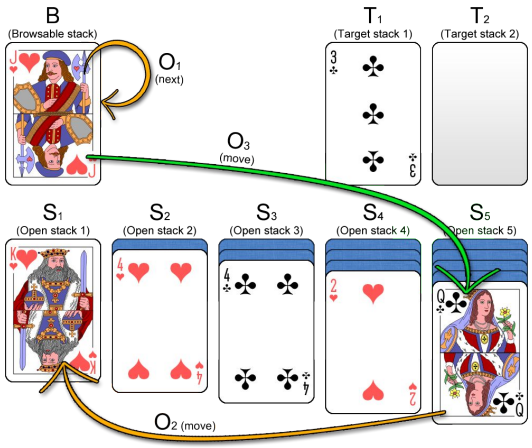
Only collaboratively playable during 1 of X phases

Collaborative Games

Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	+ Solitaire, Puzzles, Strategy and Tycoon Games, ...
Yes	- Chess, Monopoly, Worms, ...	0 Scrabble, Rummy, ...

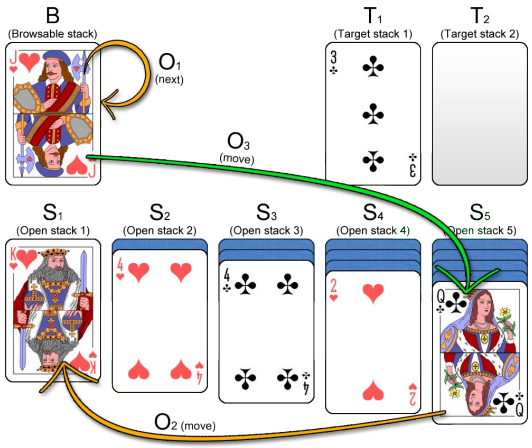
Conclusion

- **OT in games** (Example: Simplified Solitaire)



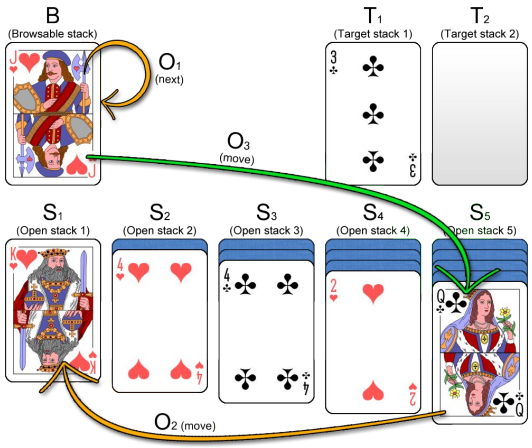
Conclusion

- **OT in games** (Example: Simplified Solitaire)
  ➡ Feasible



Conclusion

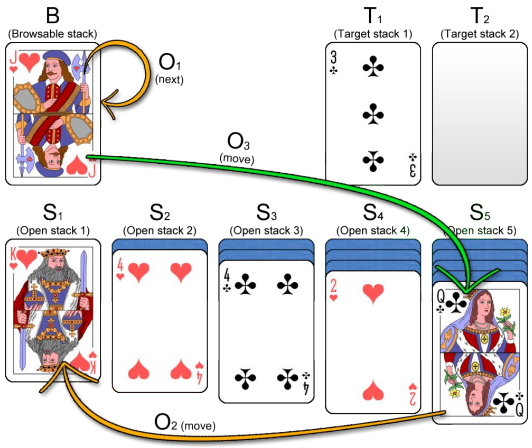
- **OT in games** (Example: Simplified Solitaire)
  ➡ Feasible
- Requires **consistency model**

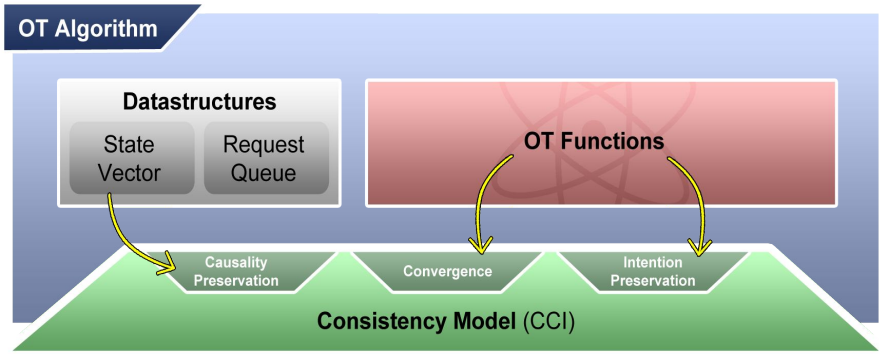


Conclusion

- **OT in games** (Example: Simplified Solitaire)
  ➡ Feasible

- Requires **consistency model**

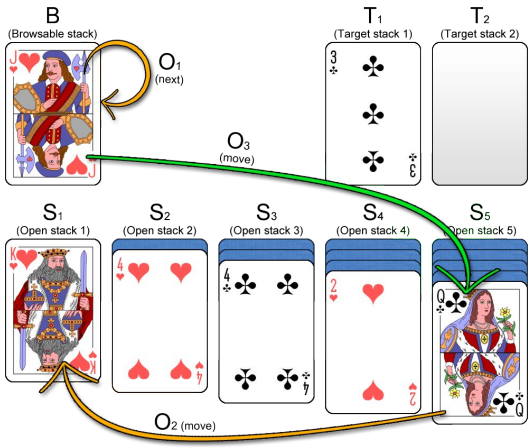




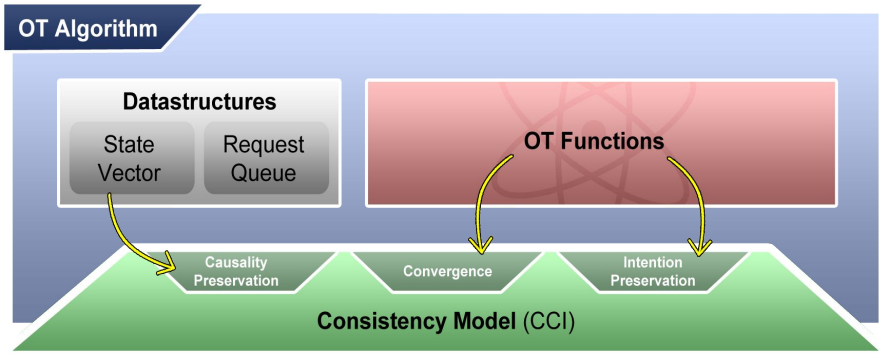
- Realized by **OT Algorithm**

Conclusion

- **OT in games** (Example: Simplified Solitaire)
  ➡ Feasible



- Requires **consistency model**



- Realized by **OT Algorithm**

- New definition of:

  Collaborative games

Turn-Based	Single-Object	Multi-Object
No	0 Pokemon, FPS, Action and Racing Games, ...	+ Solitaire, Puzzles, Strategy and Tycoon Games, ...
Yes	- Chess, Monopoly, Worms, ...	0 Scrabble, Rummy, ...

References

C. A. Ellis and S. J. Gibbs, “Concurrency control in groupware systems,” in ACM SIGMOD Record, vol. 18, pp. 399–407, ACM, 1989.

L. Lamport, “Time, clocks, and the ordering of events in a distributed system,” Communications of the ACM, vol. 21, no. 7, pp. 558–565, 1978.

D. Li and R. Li, “Preserving operation effects relation in group editors,” in Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW), pp. 457–466, ACM, 2004.

R. Li and D. Li, “A new operational transformation framework for real-time group editors,” IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 3, pp. 307–319, 2007.

R. Li and D. Li, “Commutativity-based concurrency control in groupware,” in 2005 International Conference on Collaborative Computing: Networking, Applications and Worksharing, p. 10ff., 2005.

C. Sun, Y. Zhang, X. Jia, and Y. Yang, “A generic operation transformation scheme for consistency maintenance in real-time cooperative editing systems,” in Proceedings of the international ACM SIGGROUP conference on Supporting group work: the integration challenge, pp. 425–434, ACM, 1997.

C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen, “Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems,” ACM Transactions on Computer-Human Interaction (TOCHI), vol. 5, no. 1, pp. 63–108, 1998.

D. N.-R. Matthias Ressel and R. Gunzenhuser, “An integrating, transformation-oriented approach to concurrency control and undo in group editors.,” in Proceedings of the 1996 ACM conference on Computersupported cooperative work, pp. 288–297, ACM, 1996.

S. Kumawat and A. Khuneta, “A survey on operational transformation algorithms: Challenges, issues and achievements,” International Journal of Computer Applications, vol. 3, no. 12, pp. 30–38, 2010.

J. P. Zagal, J. Rick, and I. Hsi, “Collaborative games: Lessons learned from board games,” Simul. Gaming, vol. 37, no. 1, pp. 24–40, 2006.