# CoLL

A Confluence Tool Based on Commutation

**Kiraku Shintani**
**JAIST, Japan**
s1310032@jaist.ac.jp

# Theoretical Framework

### Definition

ARSs $\mathcal{A}$ and $\mathcal{B}$ commute if

$$\mathrel{{}_{\mathcal{A}}^{*}\!\!\leftarrow} \cdot \to_{\mathcal{B}}^{*} \;\subseteq\; \to_{\mathcal{B}}^{*} \cdot \mathrel{{}_{\mathcal{A}}^{*}\!\!\leftarrow}$$

$$\begin{tikzcd}
 & \cdot \arrow[dl, "*"'] \arrow[dr, "*"] & \\
\cdot \arrow[dr, "*"'] & & \cdot \arrow[dl, "*"] \\
 & \cdot &
\end{tikzcd}$$

# Theoretical Framework

### Definition

ARSs $\mathcal{A}$ and $\mathcal{B}$ commute if

$${}_{\mathcal{A}}^{*}\!\leftarrow \cdot \rightarrow_{\mathcal{B}}^{*} \subseteq \rightarrow_{\mathcal{B}}^{*} \cdot {}_{\mathcal{A}}^{*}\!\leftarrow$$

$$\begin{tikzcd}
& \cdot \arrow[dl, "*"'] \arrow[dr, "*"] & \\
\cdot \arrow[dr, "*"'] & & \cdot \arrow[dl, "*"] \\
& \cdot &
\end{tikzcd}$$

### Theorem (Commutation Lemma [Hindley 64])

*ARSs $\mathcal{A}$ and $\mathcal{B}$ commute if every $\rightarrow_\alpha$ and $\rightarrow_\beta$ commute*

*where* $\mathcal{A} = \langle A, \{\rightarrow_\alpha\}_{\alpha \in I} \rangle$ *and* $\mathcal{B} = \langle A, \{\rightarrow_\beta\}_{\beta \in J} \rangle$

# Theoretical Framework

### Definition

ARSs $\mathcal{A}$ and $\mathcal{B}$ commute if

$${}_{\mathcal{A}}^{*}\!\leftarrow \cdot \to_{\mathcal{B}}^{*} \subseteq \to_{\mathcal{B}}^{*} \cdot {}_{\mathcal{A}}^{*}\!\leftarrow$$

$$\begin{tikzcd}
& \cdot \arrow[dl, "*"'] \arrow[dr, "*"] & \\
\cdot \arrow[dr, "*"'] & & \cdot \arrow[dl, "*"] \\
& \cdot &
\end{tikzcd}$$

### Theorem (Commutation Lemma [Hindley 64])

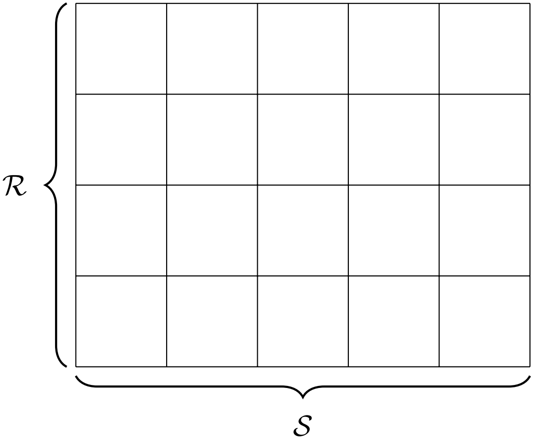
*ARSs $\mathcal{A}$ and $\mathcal{B}$ commute if every $\to_\alpha$ and $\to_\beta$ commute*

*where* $\mathcal{A} = \langle A, \{\to_\alpha\}_{\alpha \in I} \rangle$ *and* $\mathcal{B} = \langle A, \{\to_\beta\}_{\beta \in J} \rangle$

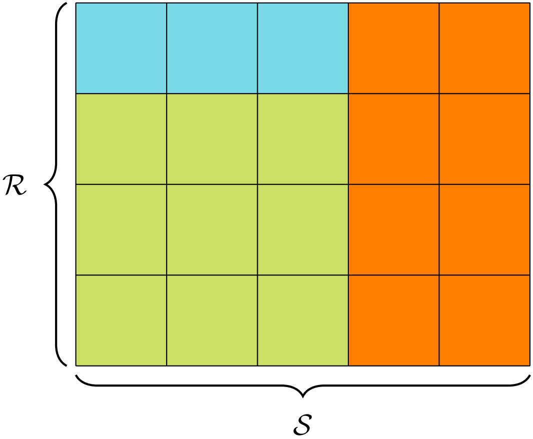
### Lemma (self commutation)

*ARS $\mathcal{A}$ is confluent if $\mathcal{A}$ and $\mathcal{A}$ commute*

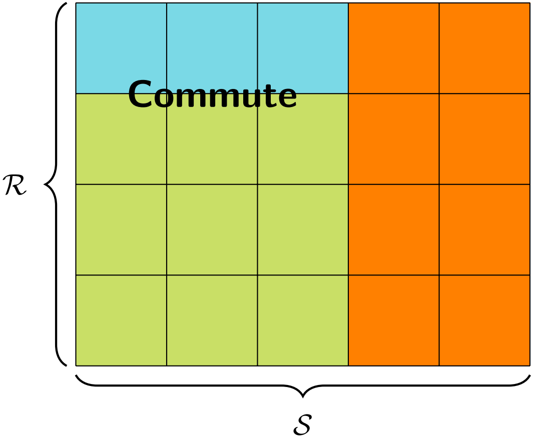
# Commutation Lemma



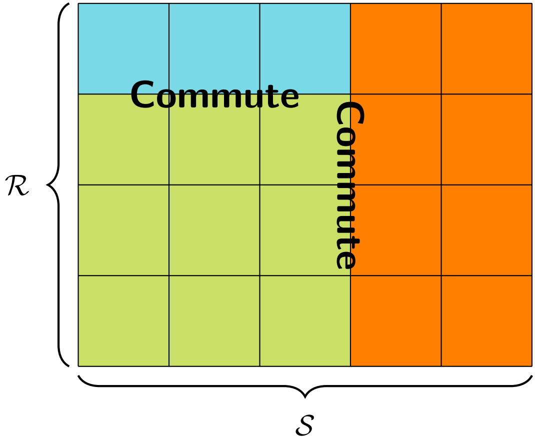
# Commutation Lemma



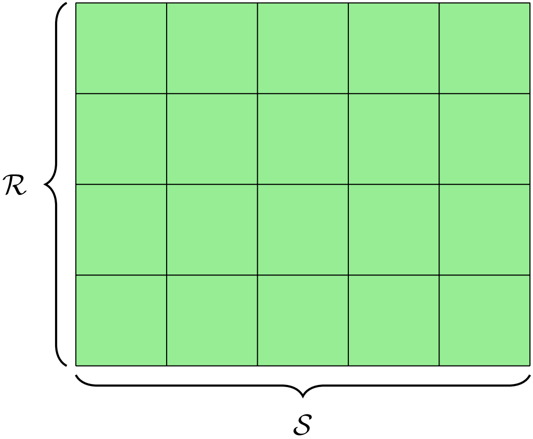
# Commutation Lemma



# Commutation Lemma

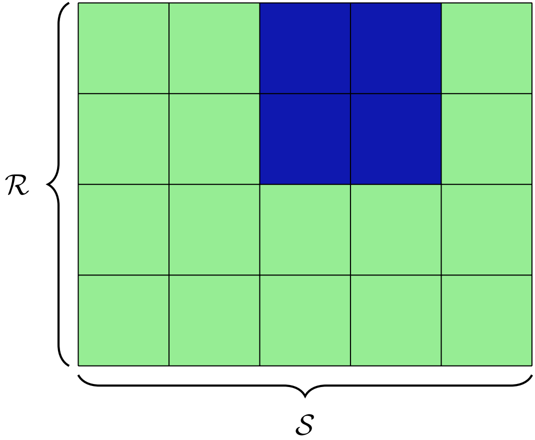


# Commutation Lemma



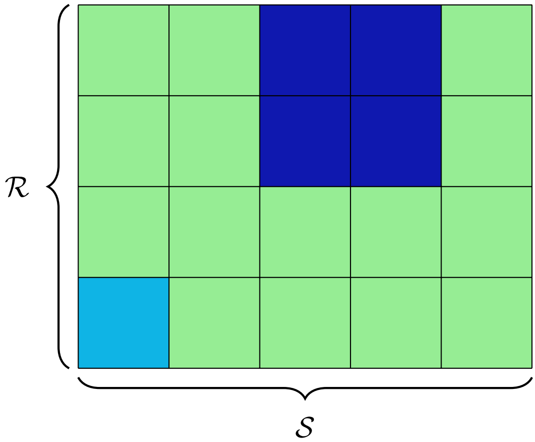
$\mathcal{R}$ and $\mathcal{S}$ commute

# Commutation Lemma



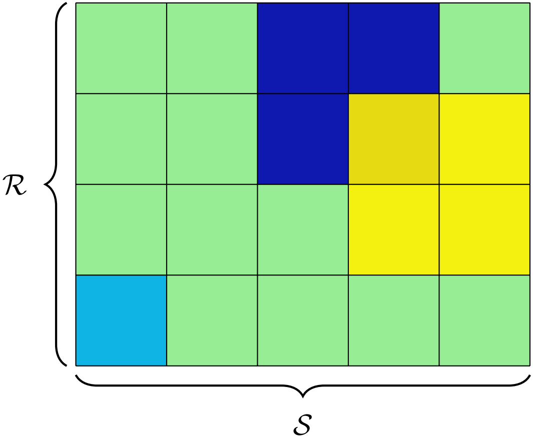
$\mathcal{R}$ and $\mathcal{S}$ commute . . .

# Commutation Lemma



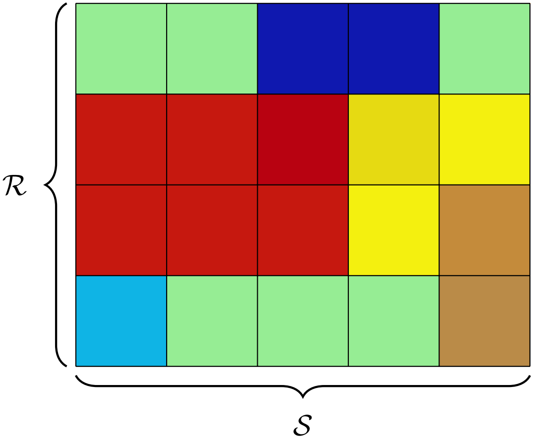
$\mathcal{R}$ and $\mathcal{S}$ commute . . .

# Commutation Lemma



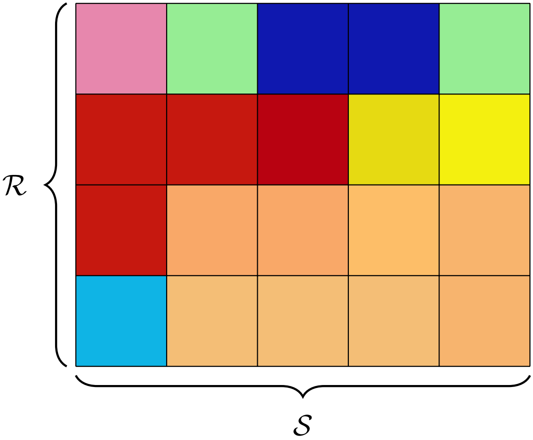
$\mathcal{R}$ and $\mathcal{S}$ commute . . .

# Commutation Lemma



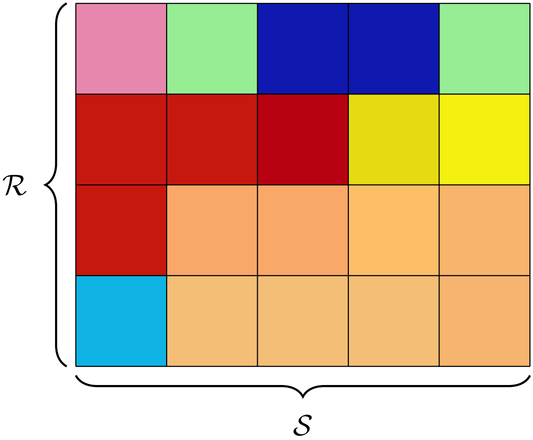
$\mathcal{R}$ and $\mathcal{S}$ commute . . .

# Commutation Lemma



too many commutable subsystems!

# Commutation Lemma

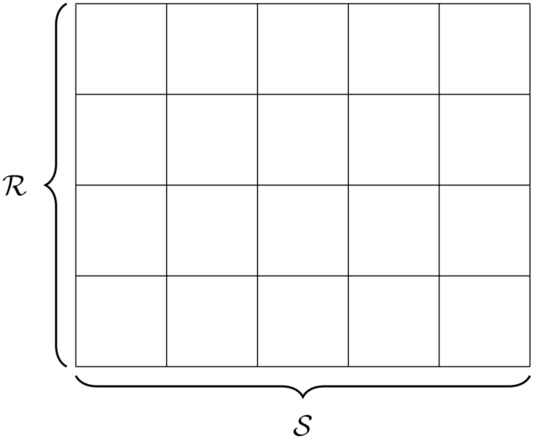


too many commutable subsystems!

how to find suitable subsystems?

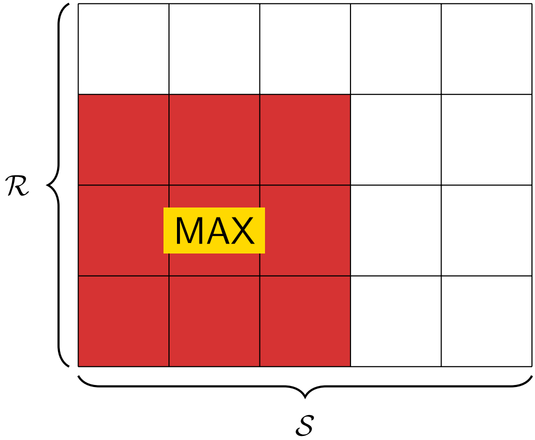
# MaxSAT

every time, take maximal commuting subsystems



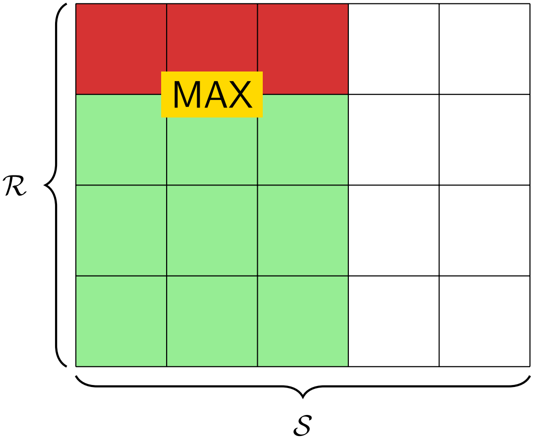
# MaxSAT

every time, take maximal commuting subsystems



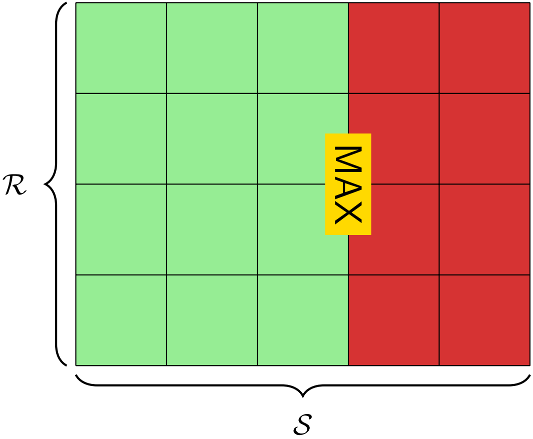
# MaxSAT

every time, take maximal commuting subsystems



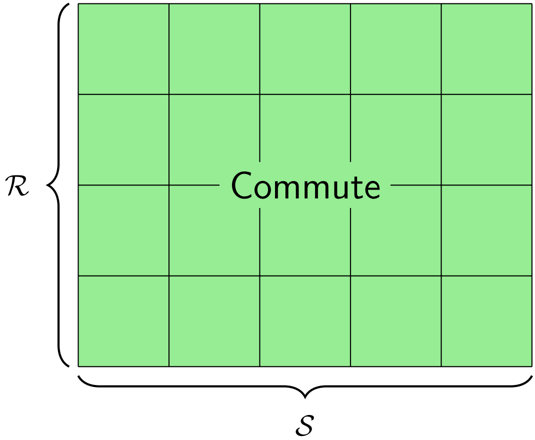
# MaxSAT

every time, take maximal commuting subsystems



# MaxSAT

every time, take maximal commuting subsystems



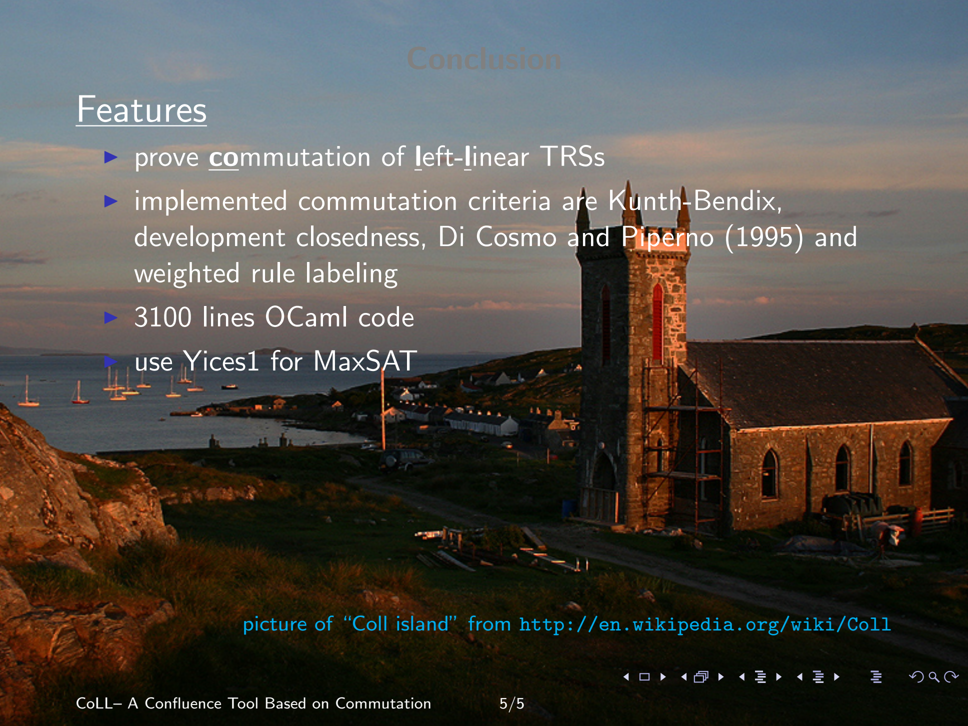
# Conclusion

picture of “Coll island” from http://en.wikipedia.org/wiki/Coll

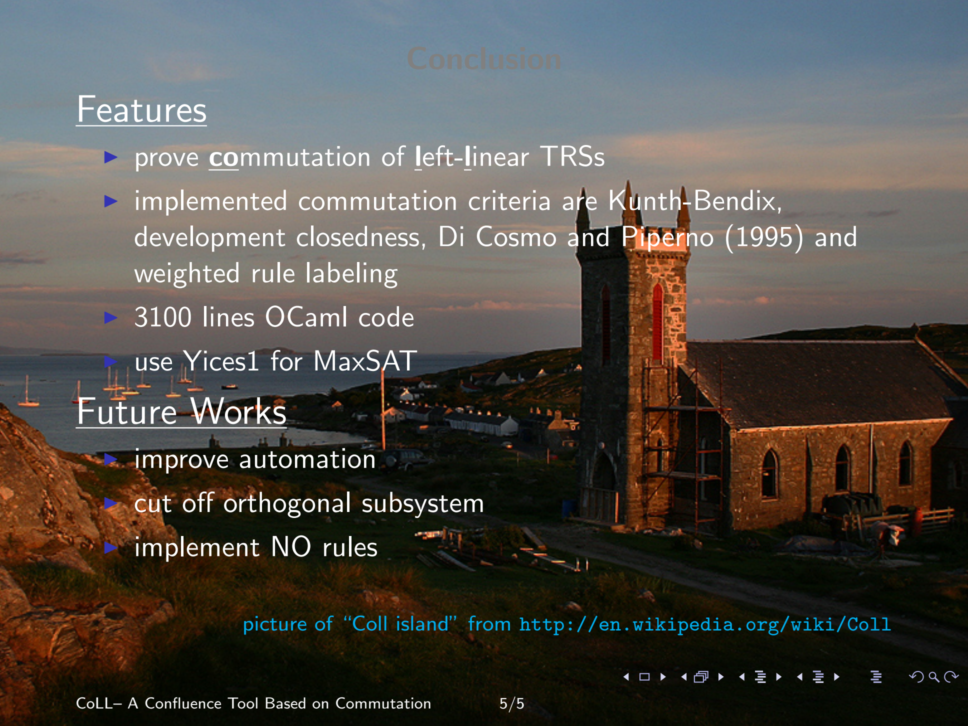
# Conclusion

## Features

- prove **co**mmutation of **l**eft-**l**inear TRSs
- implemented commutation criteria are Kunth-Bendix, development closedness, Di Cosmo and Piperno (1995) and weighted rule labeling
- 3100 lines OCaml code
- use Yices1 for MaxSAT



picture of "Coll island" from http://en.wikipedia.org/wiki/Coll



# Conclusion

## Features

- prove **co**mmutation of **l**eft-**l**inear TRSs
- implemented commutation criteria are Kunth-Bendix, development closedness, Di Cosmo and Piperno (1995) and weighted rule labeling
- 3100 lines OCaml code
- use Yices1 for MaxSAT

## Future Works

- improve automation
- cut off orthogonal subsystem
- implement NO rules

picture of "Coll island" from `http://en.wikipedia.org/wiki/Coll`