TFNP

TBD

Proof Complexity, Circuit Complexity, and TFNP

Noah Fleming
Memorial University

Based on work with Sam Buss and Russell Impagliazzo

Monotone Circuit Complexity

Task

Object

Model

Monotone
Circuit Model M

Monotone Circuit Complexity

Task

Object

Model

Monotone Circuit Model M

Monotone Circuit Complexity

Task

Object

Model

Monotone Function f

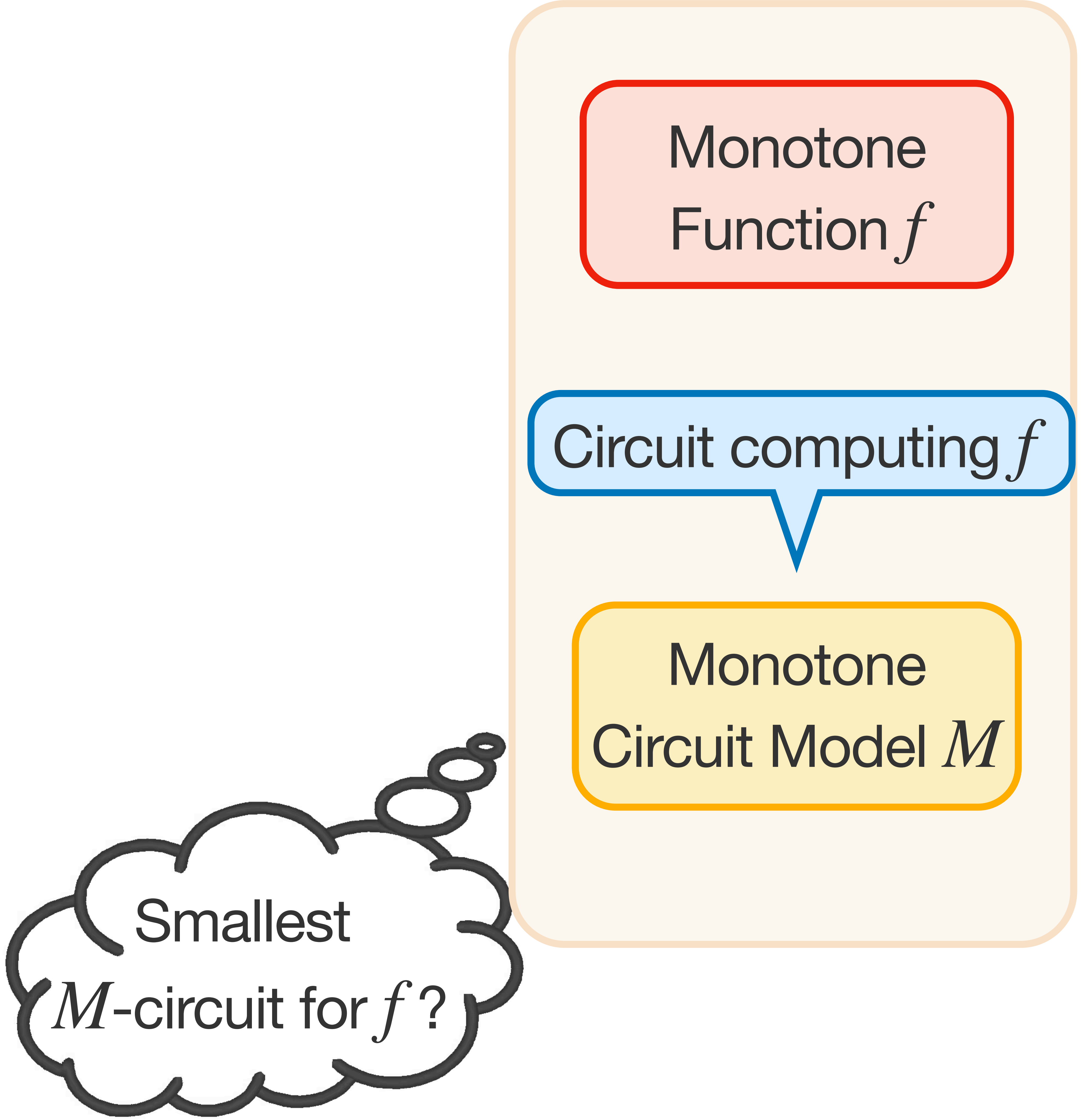
Monotone Circuit Model M

Monotone Circuit Complexity

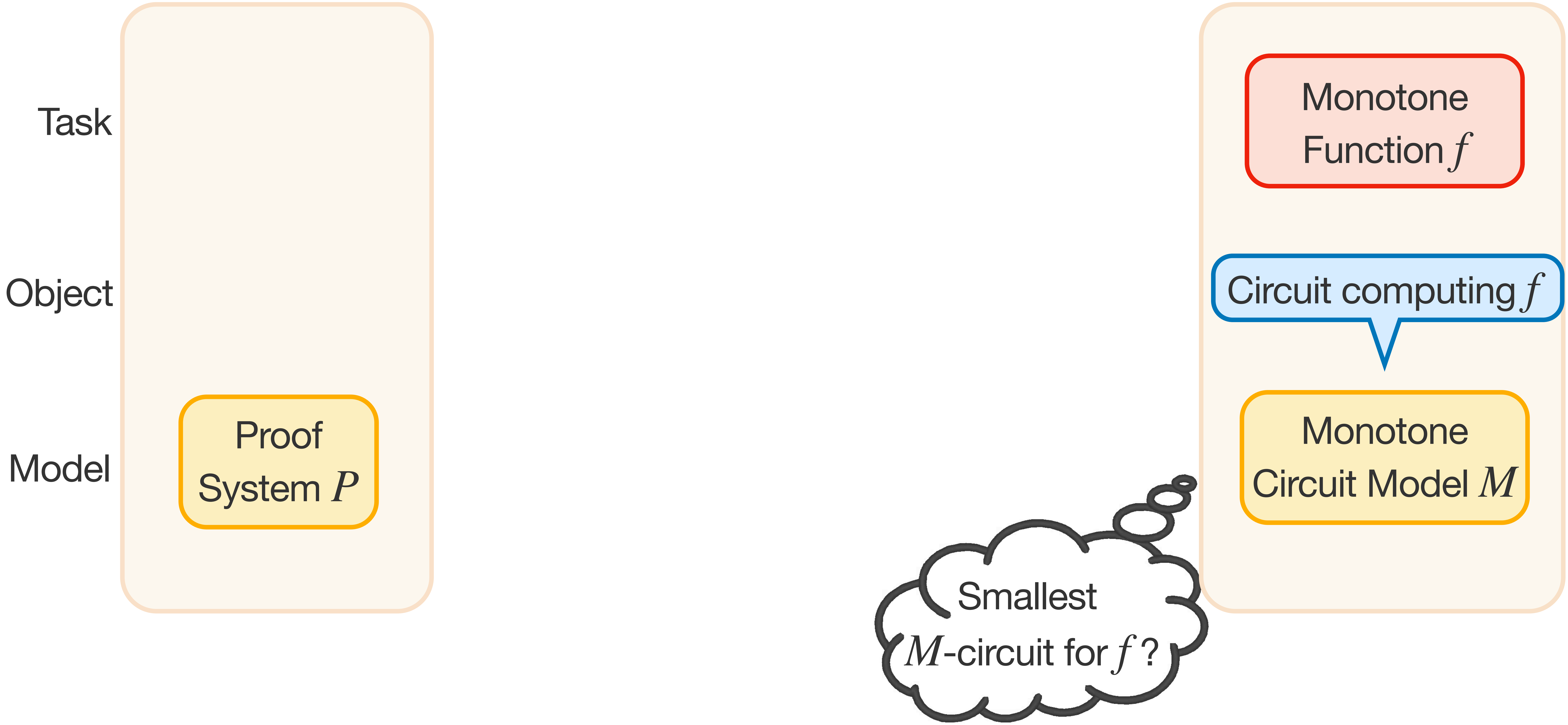
Task

Object

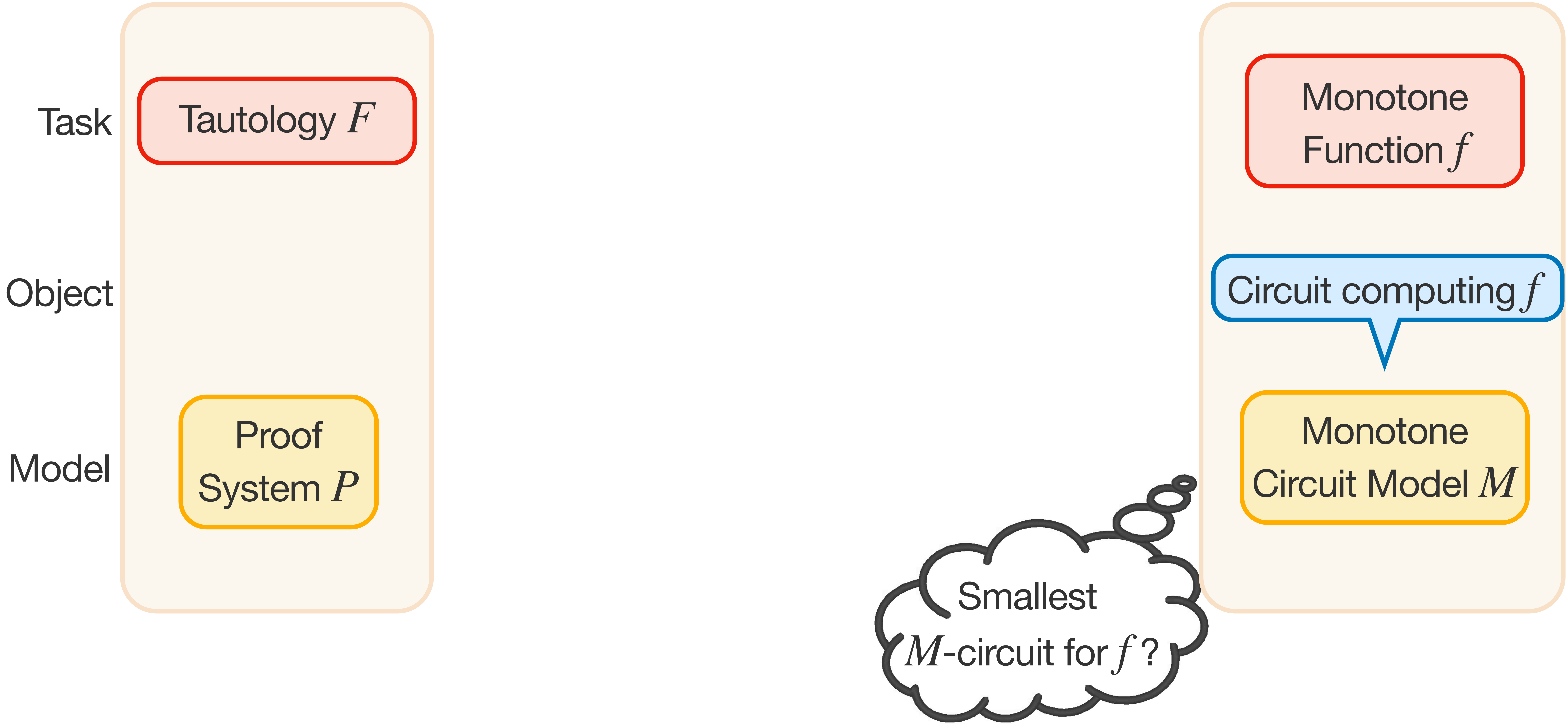
Model



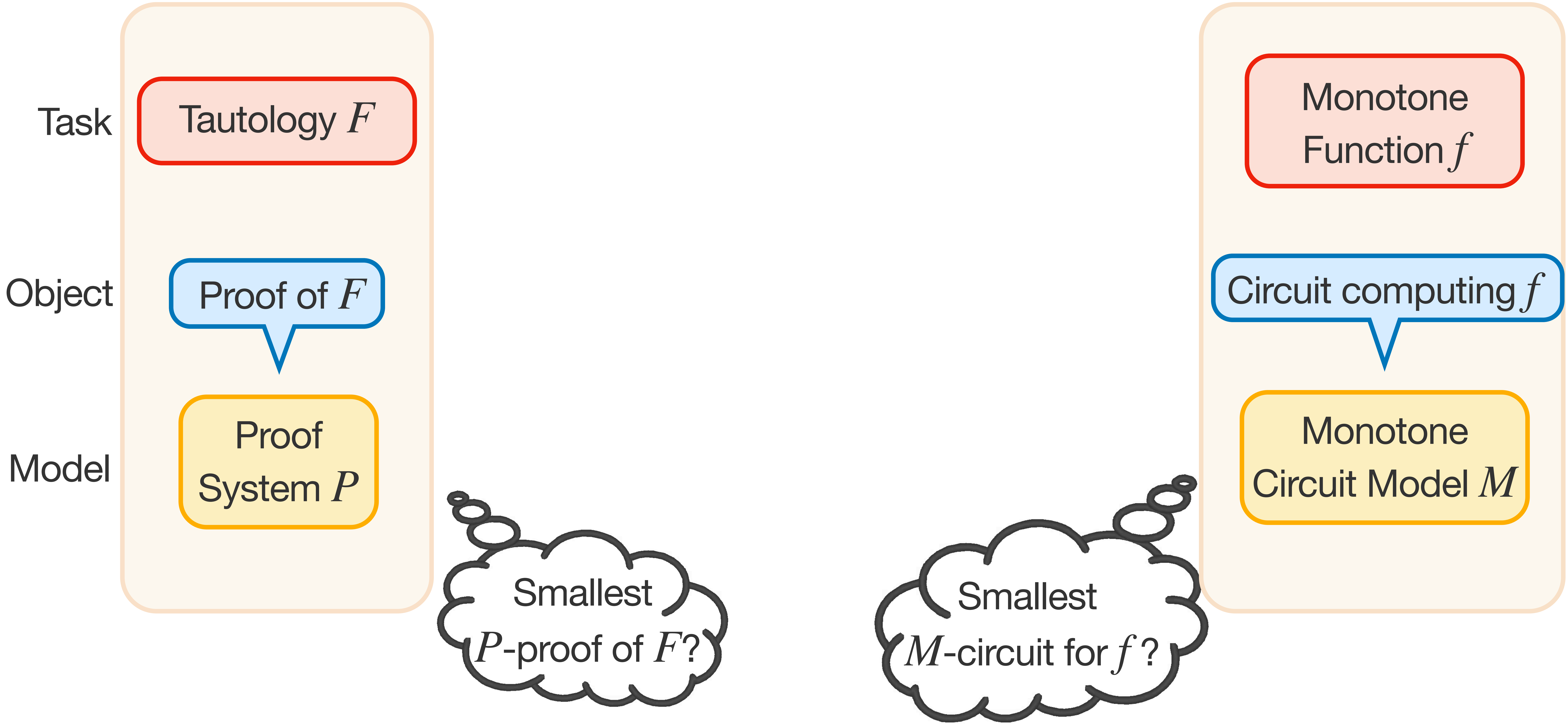
Proof Complexity



Proof Complexity



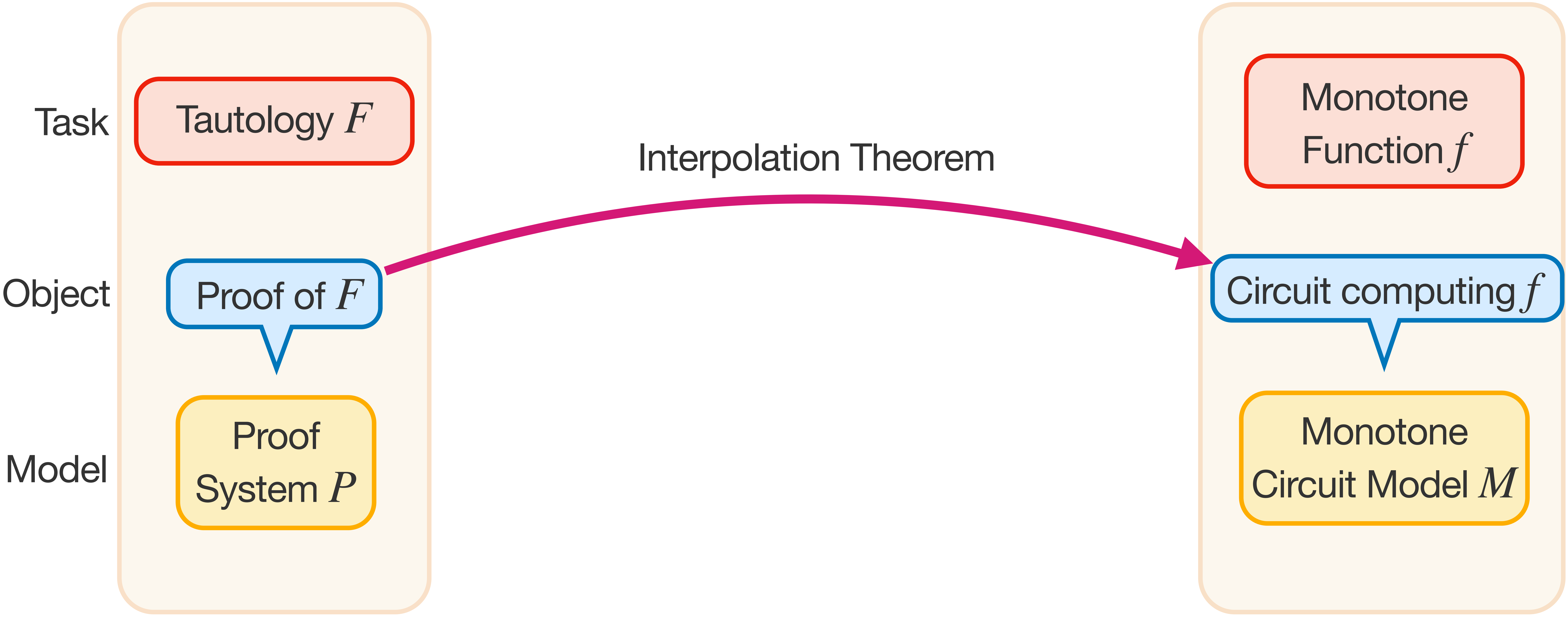
Proof Complexity



Interplay

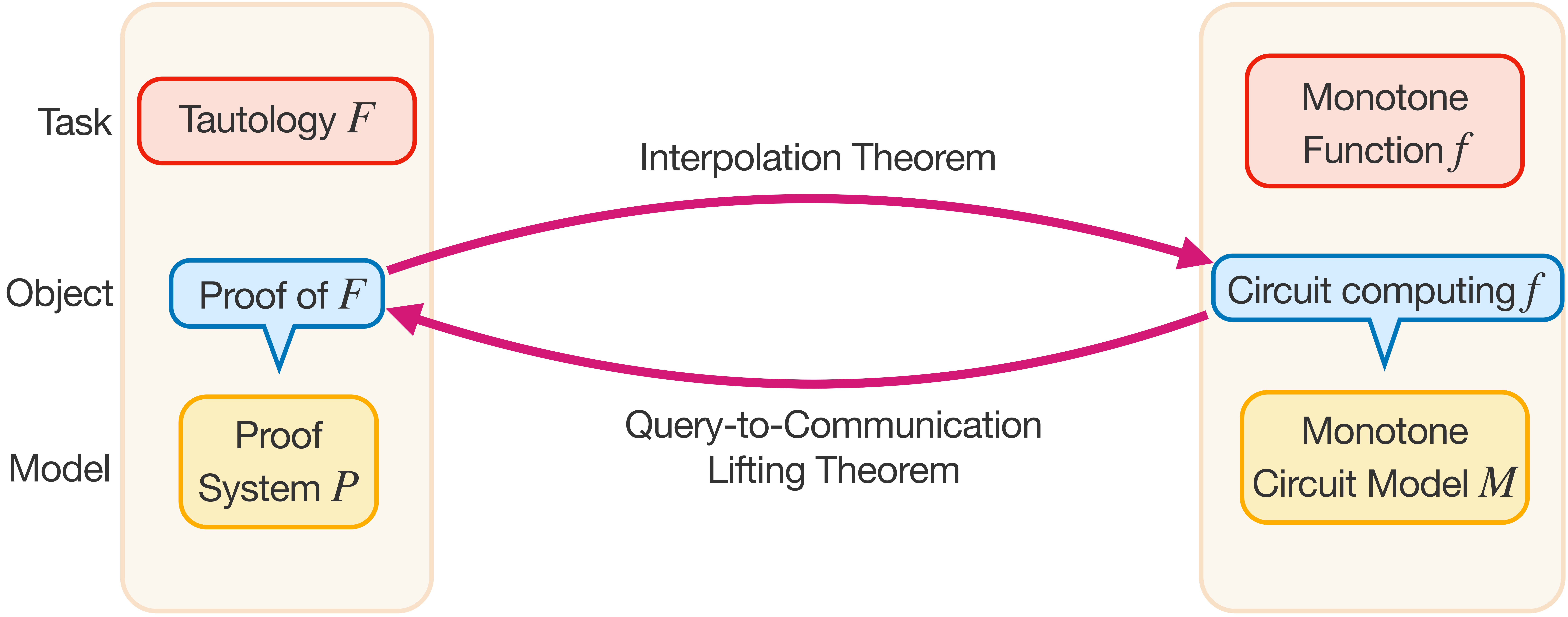
Major breakthroughs resulted from uncovering deep connections between these areas!

Interplay



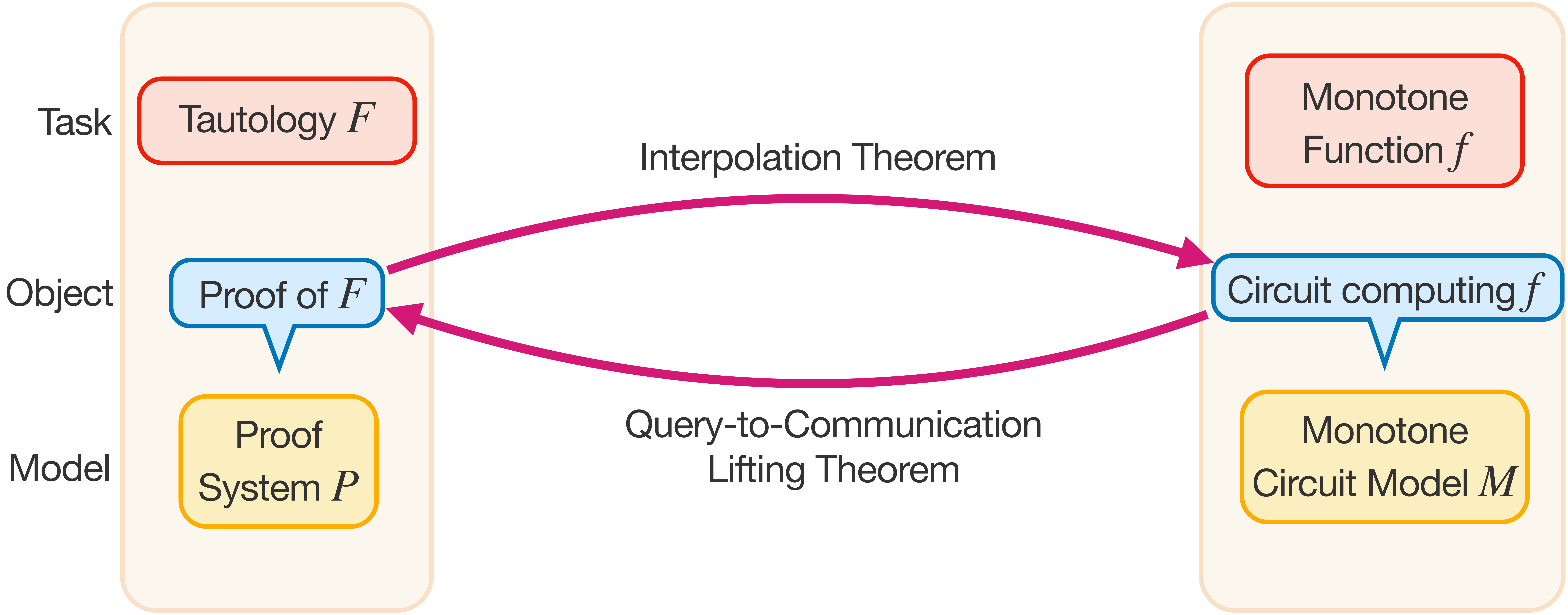
Major breakthroughs resulted from uncovering deep connections between these areas!

Interplay



Major breakthroughs resulted from uncovering deep connections between these areas!

Interplay



Major breakthroughs resulted from uncovering deep connections between these areas!

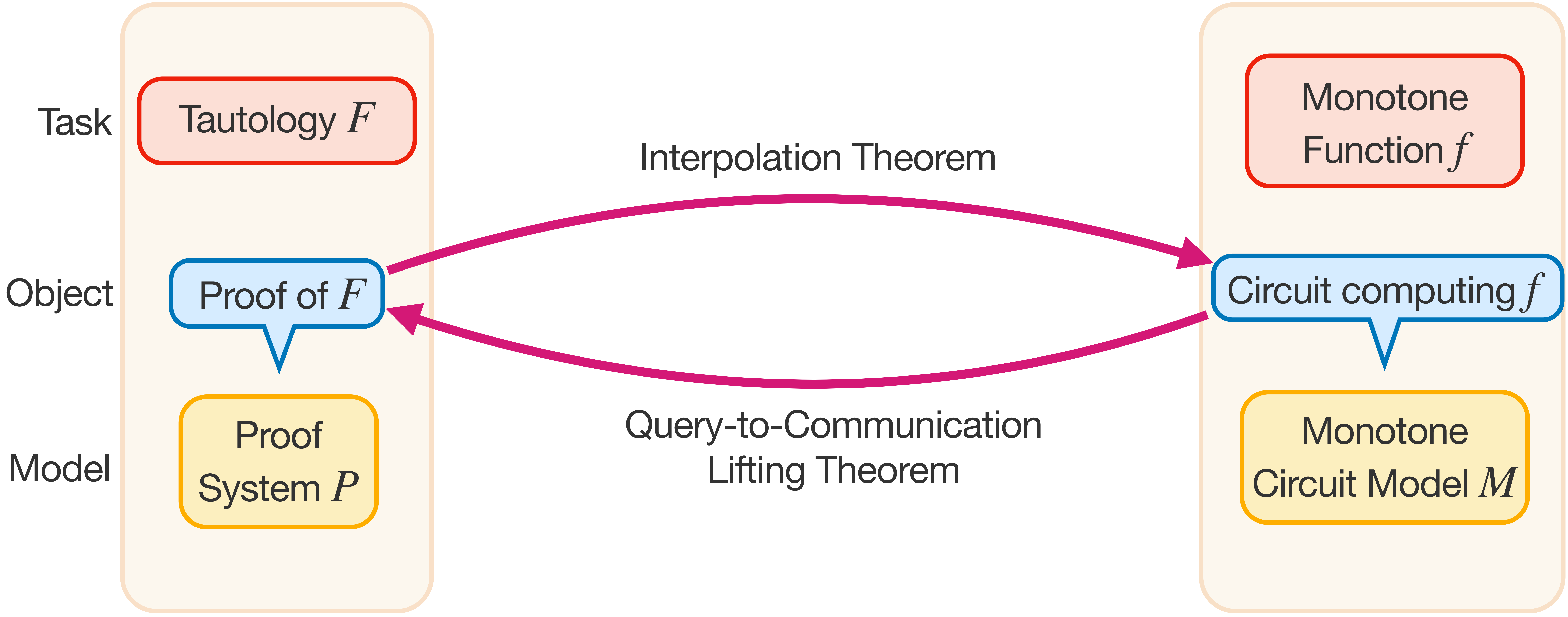
Upshot: Tools from one area can be applied to the other!

Interplay

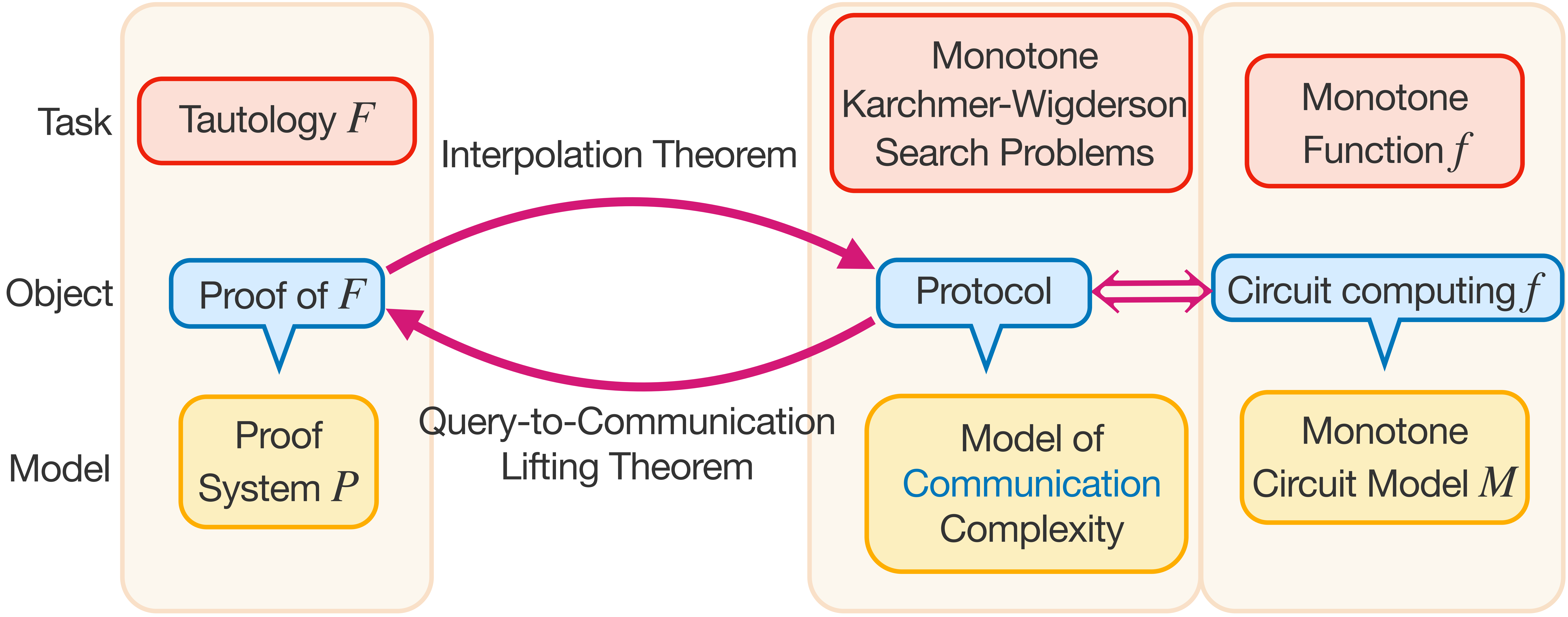
Q. When and why do these connections occur?

TFNP has emerged as a roadmap for interpolation and lifting theorems

Characterizations by Total Search Problems

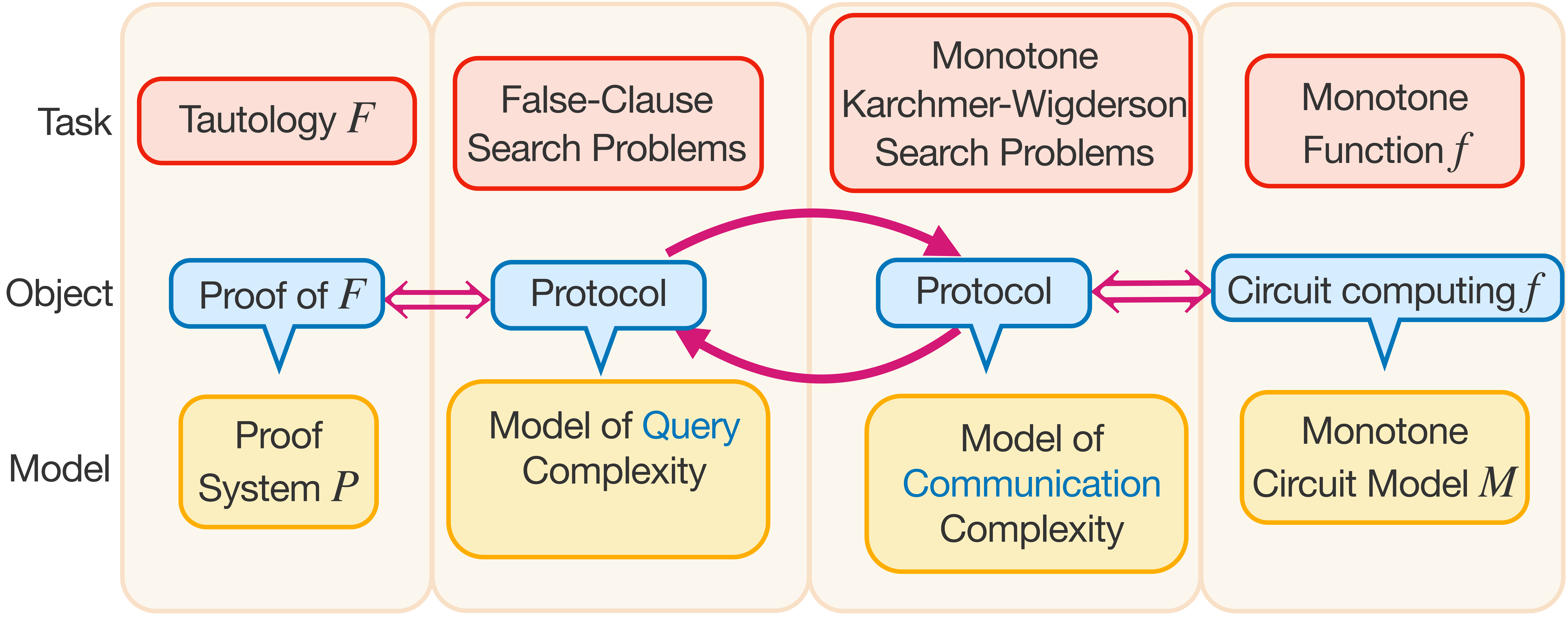


Characterizations by Total Search Problems



mKW_f: Given $(x, y) \in f^{-1}(1) \times f^{-1}(0)$ output $i \in [n]$ such that $x_i \neq y_i$

Characterizations by Total Search Problems



$Search_F$: Given $x \in \{0,1\}^n$ output the index of a clause of F falsified by x

TFNP

Studies the complexity of computing total search problems

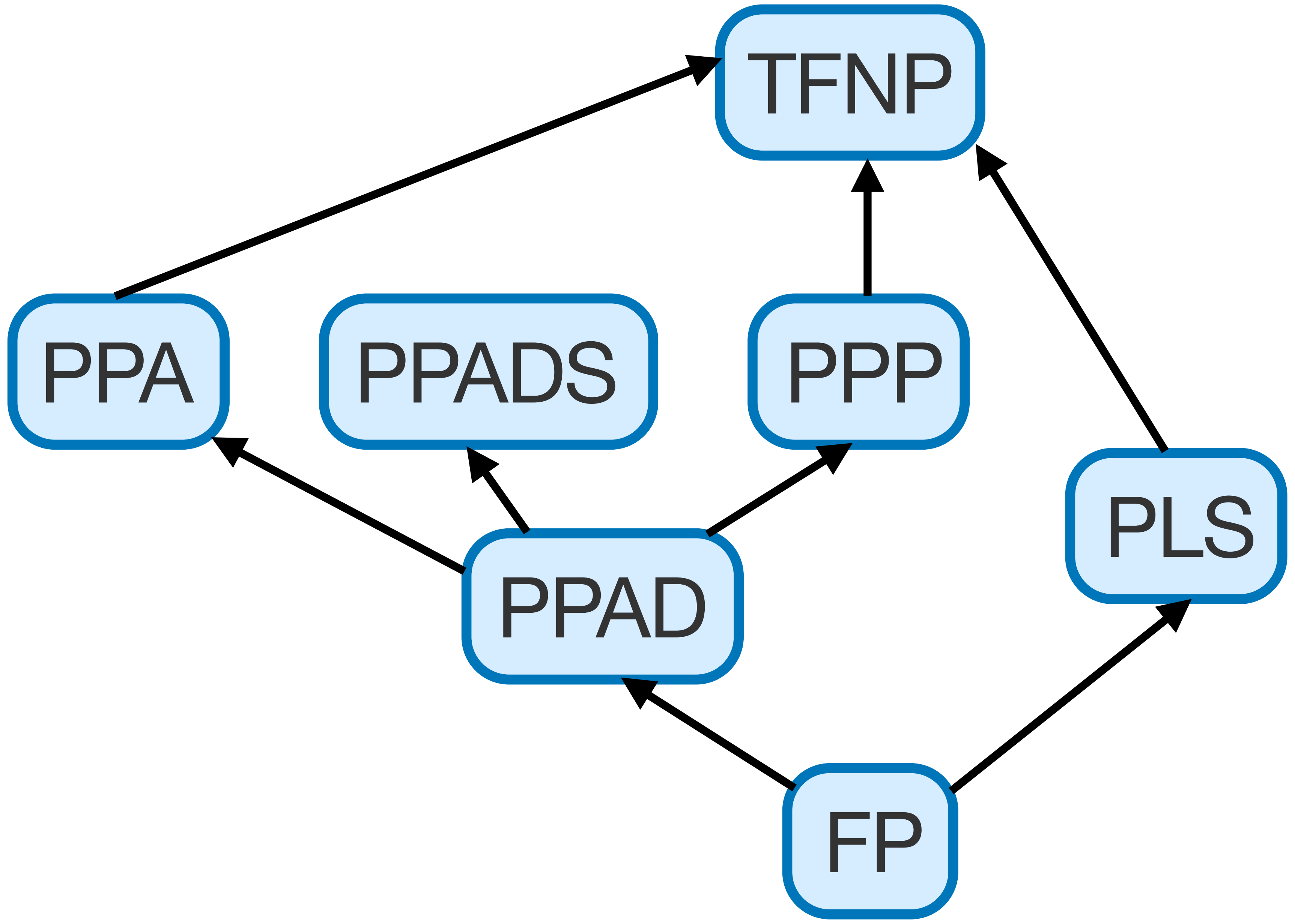
TFNP

Studies the complexity of computing total search problems

→Organizes them into a variety of classes with complete problems

TFNP

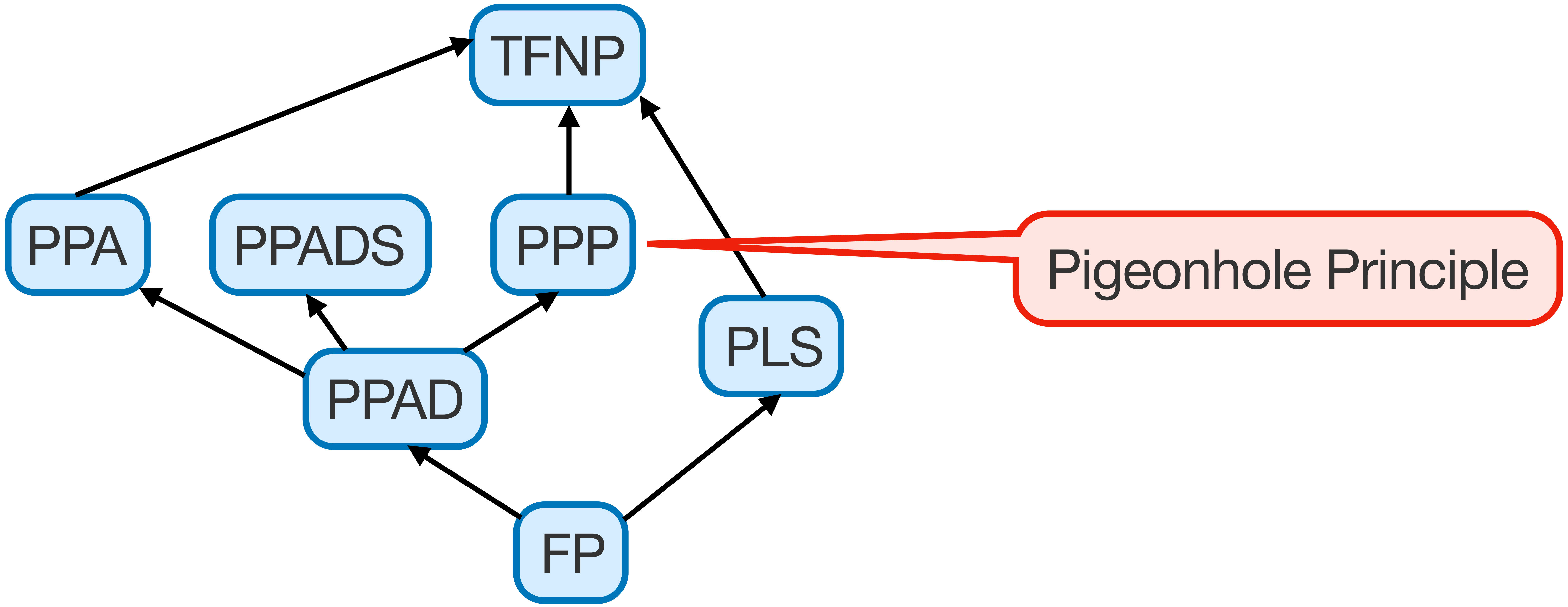
Studies the complexity of computing total search problems
→Organizes them into a variety of classes with complete problems



TFNP

Studies the complexity of computing total search problems

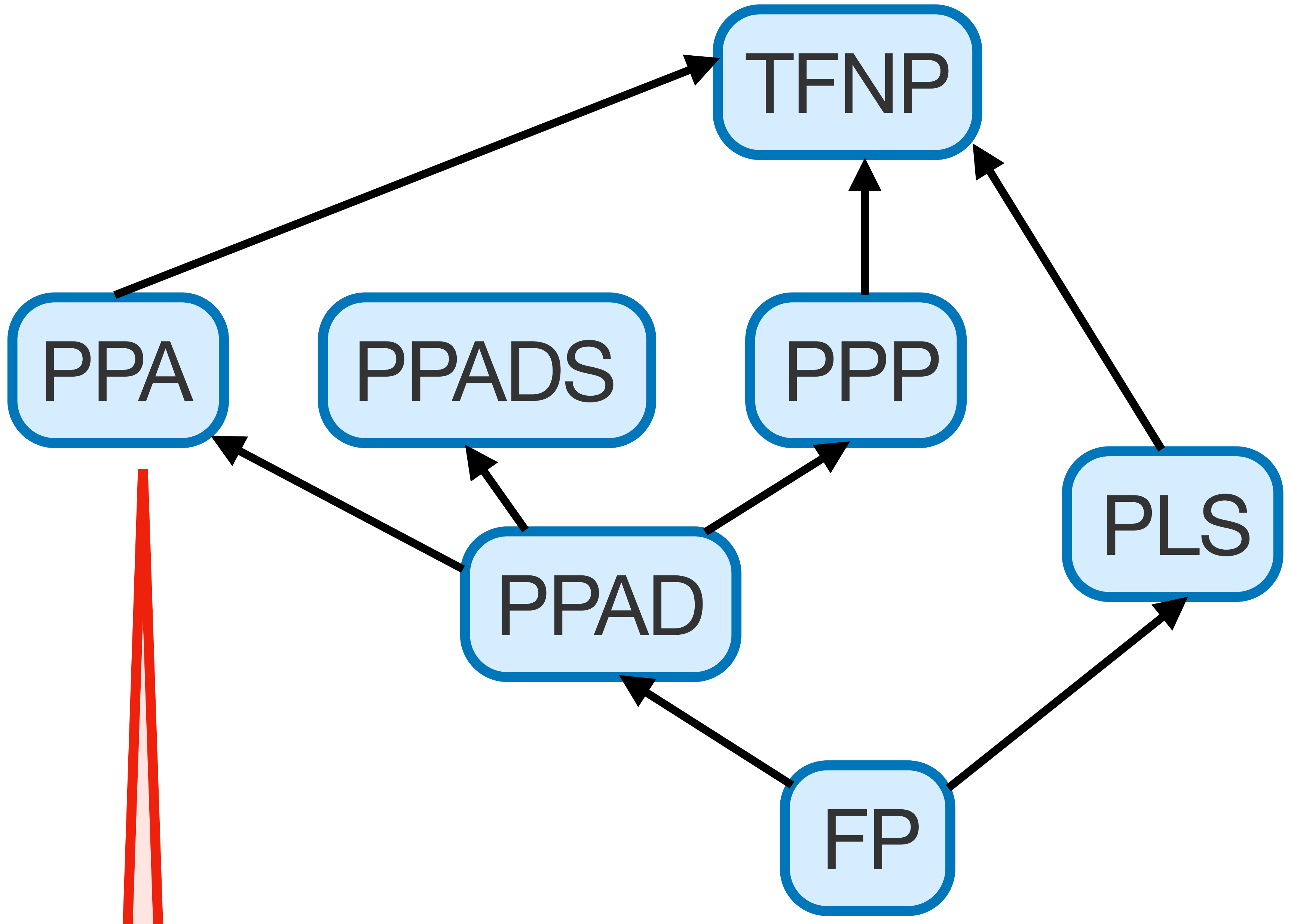
→Organizes them into a variety of classes with complete problems



TFNP

Studies the complexity of computing total search problems

→Organizes them into a variety of classes with complete problems

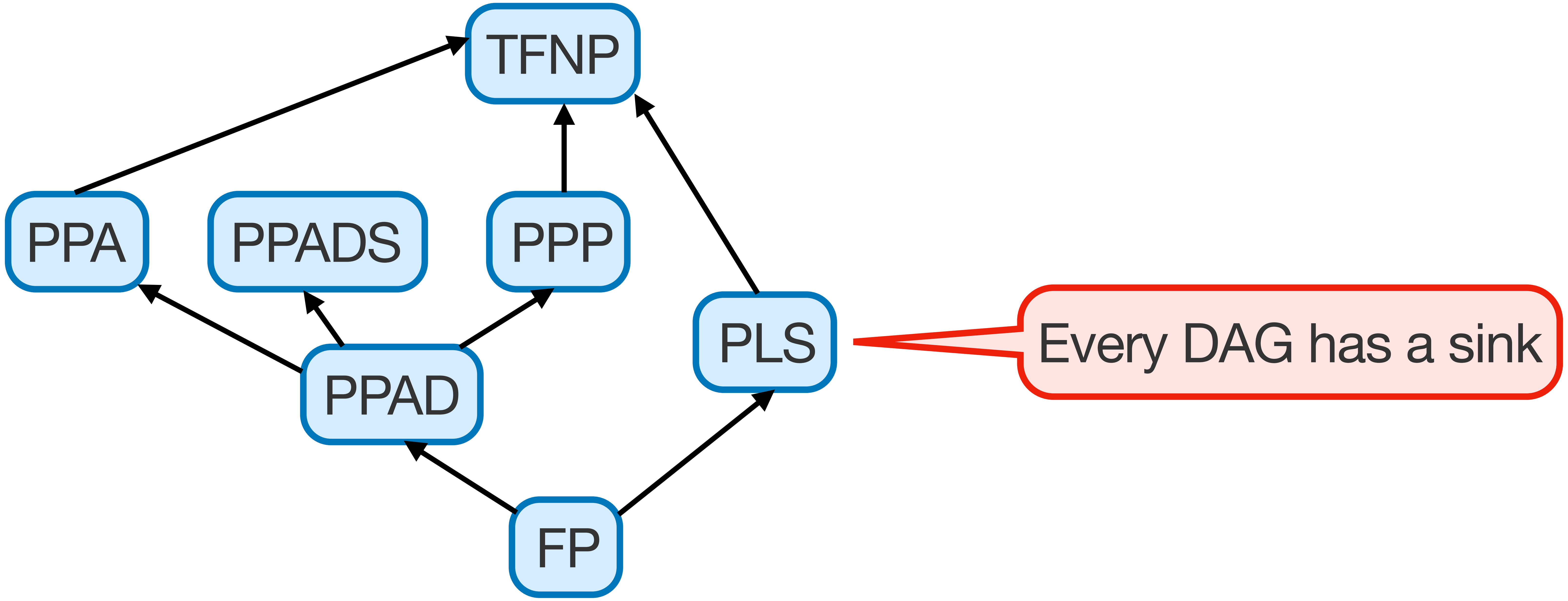


Every odd degree vertex has another

TFNP

Studies the complexity of computing total search problems

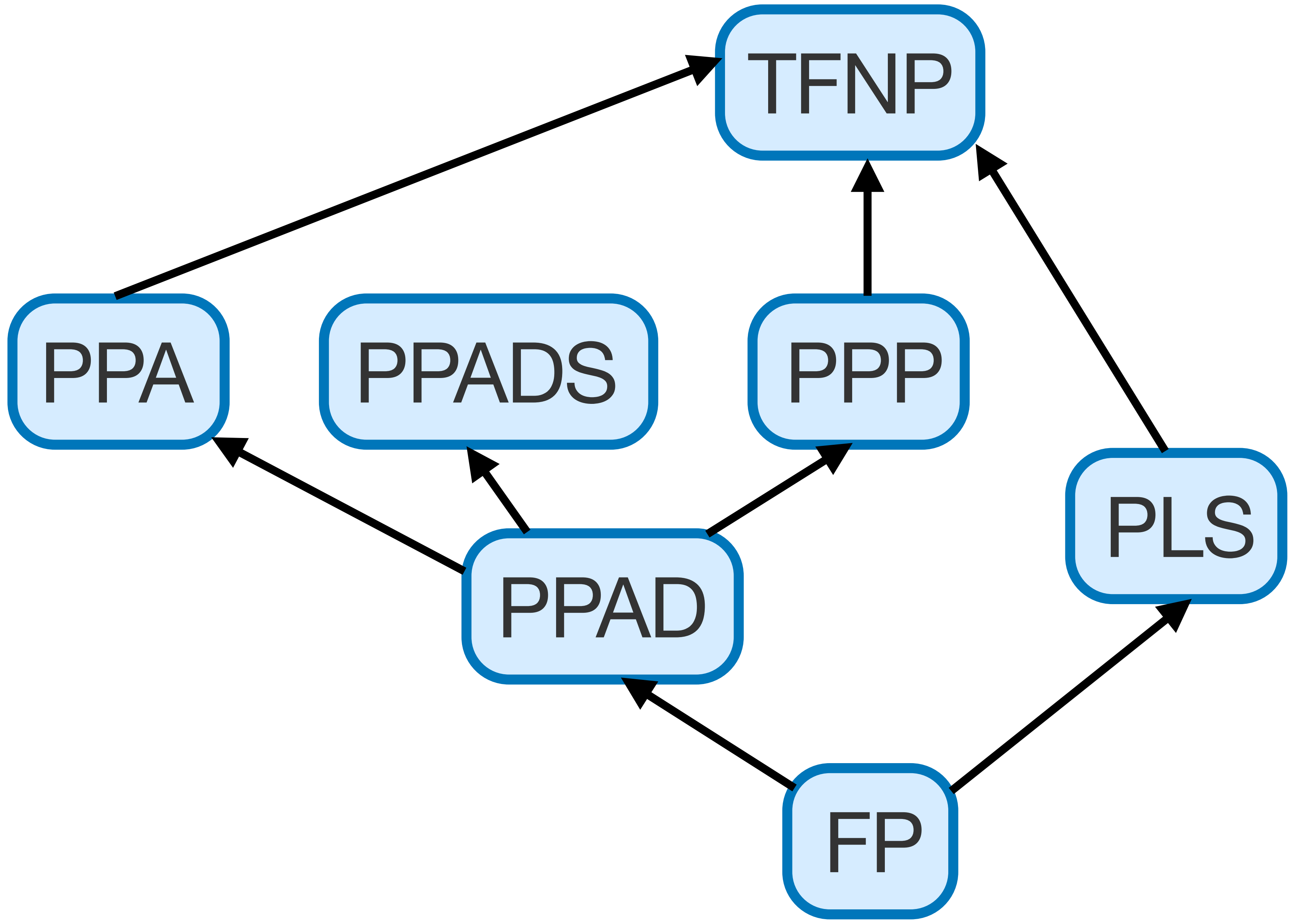
→Organizes them into a variety of classes with complete problems



TFNP

Studies the complexity of computing total search problems

→Organizes them into a variety of classes with complete problems



Every DAG has a sink

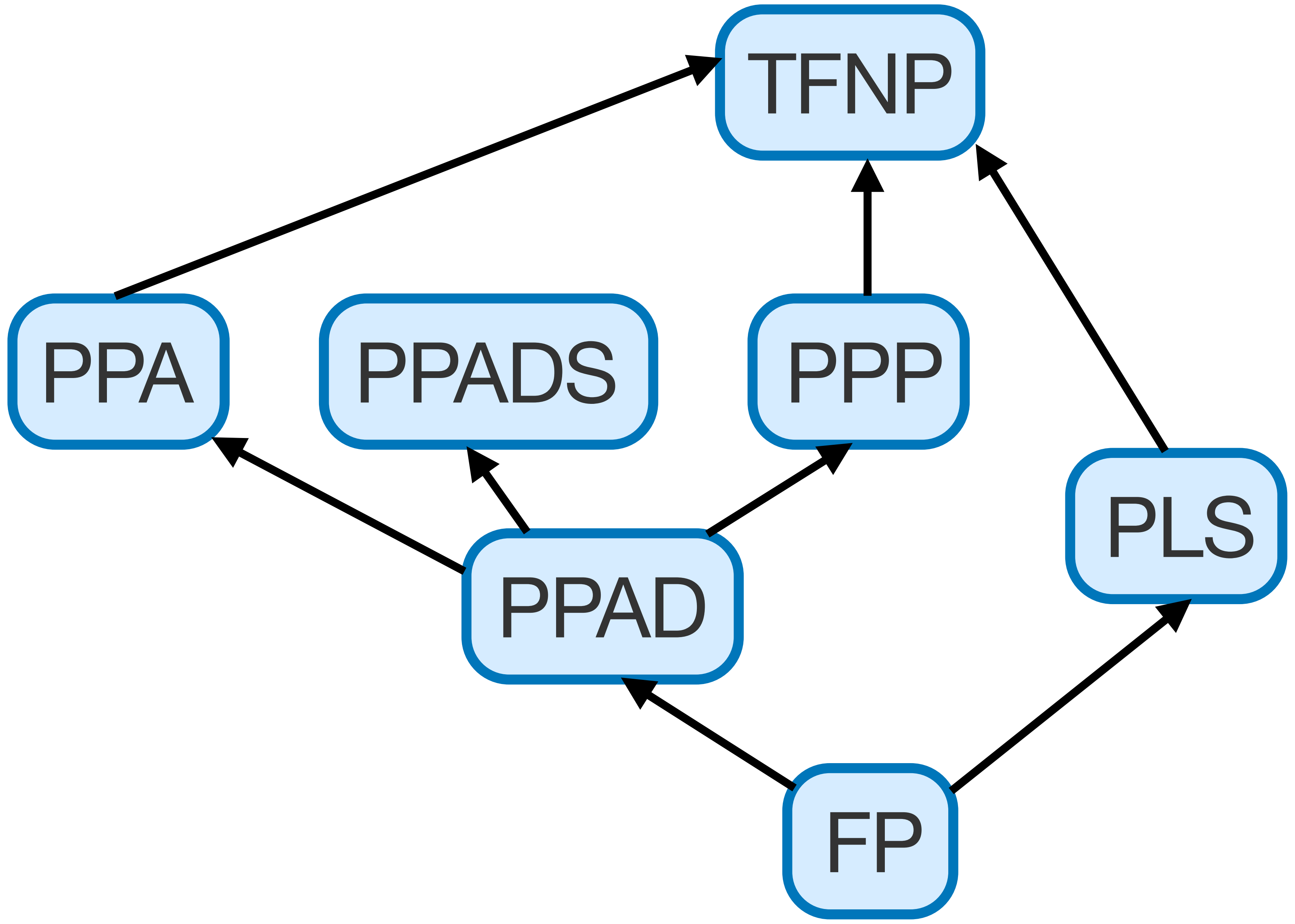
SinkOfDag

Vertices: $1, \ldots, n$

TFNP

Studies the complexity of computing total search problems

→Organizes them into a variety of classes with complete problems

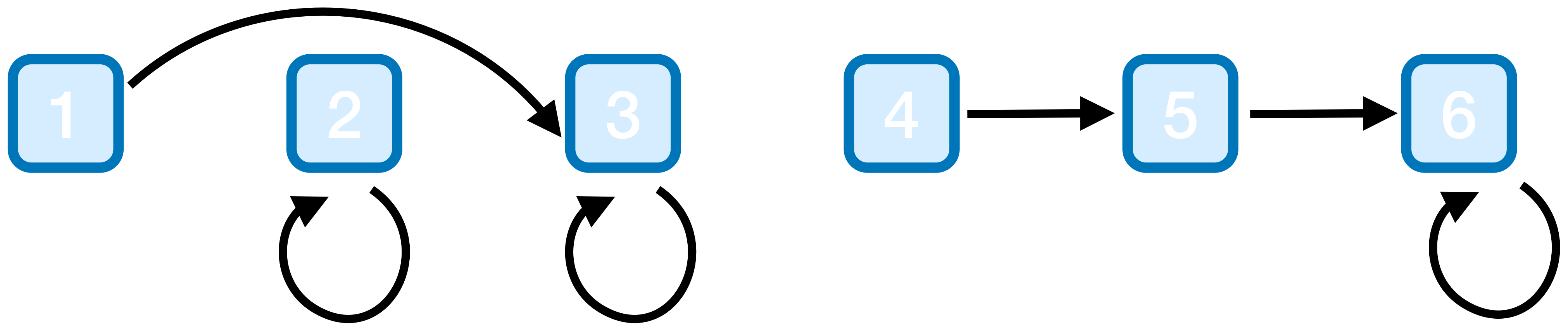


Every DAG has a sink

SinkOfDag

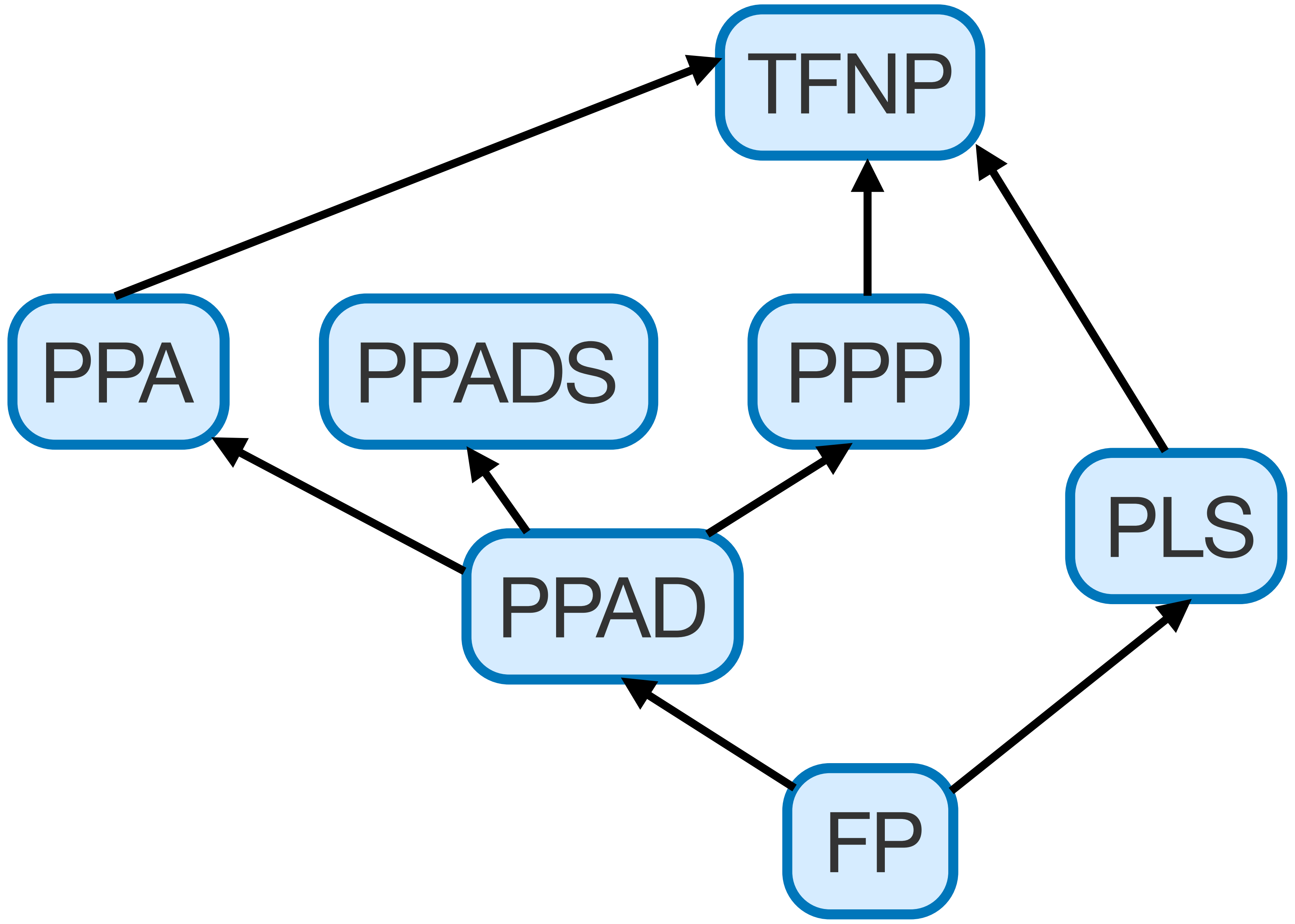
Vertices: $1, \ldots, n$

Successor pointers: $s_i \geq i$ with $s_1 \neq 1$



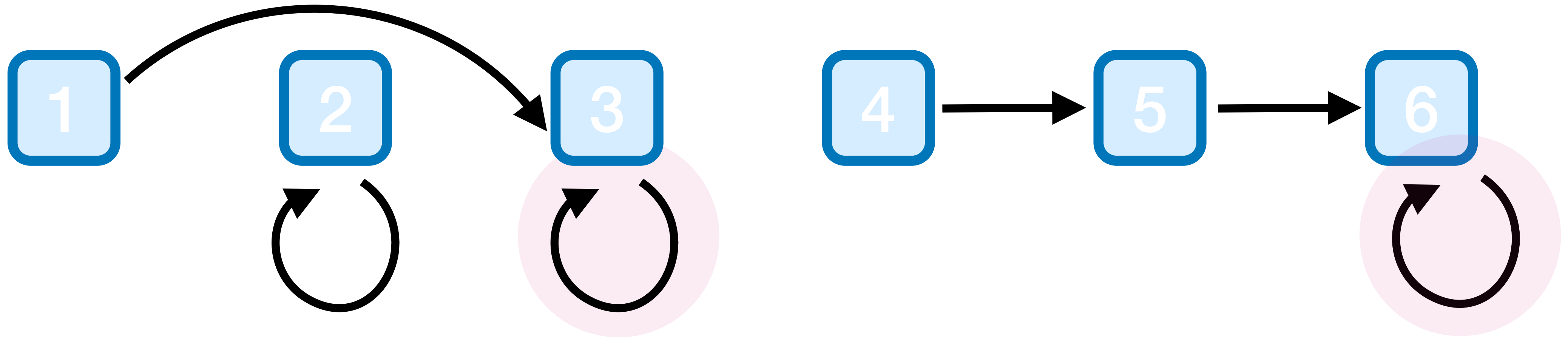
TFNP

Studies the complexity of computing total search problems
→Organizes them into a variety of classes with complete problems



Every DAG has a sink

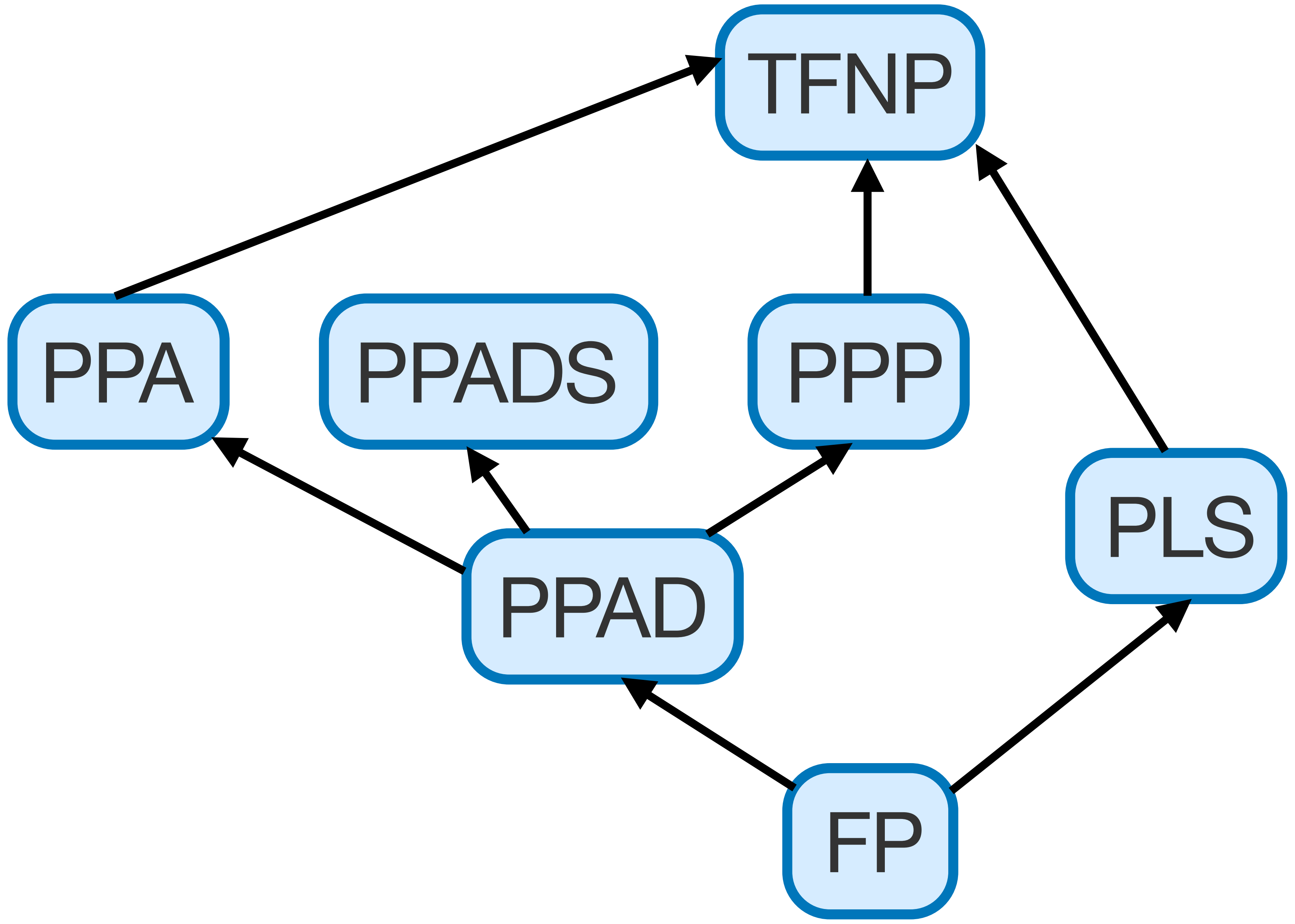
SinkOfDag
Vertices: $1, \ldots, n$
Successor pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i such that $s_i \neq i$ but $s_{s_i} = s_i$



TFNP

Studies the complexity of computing total search problems

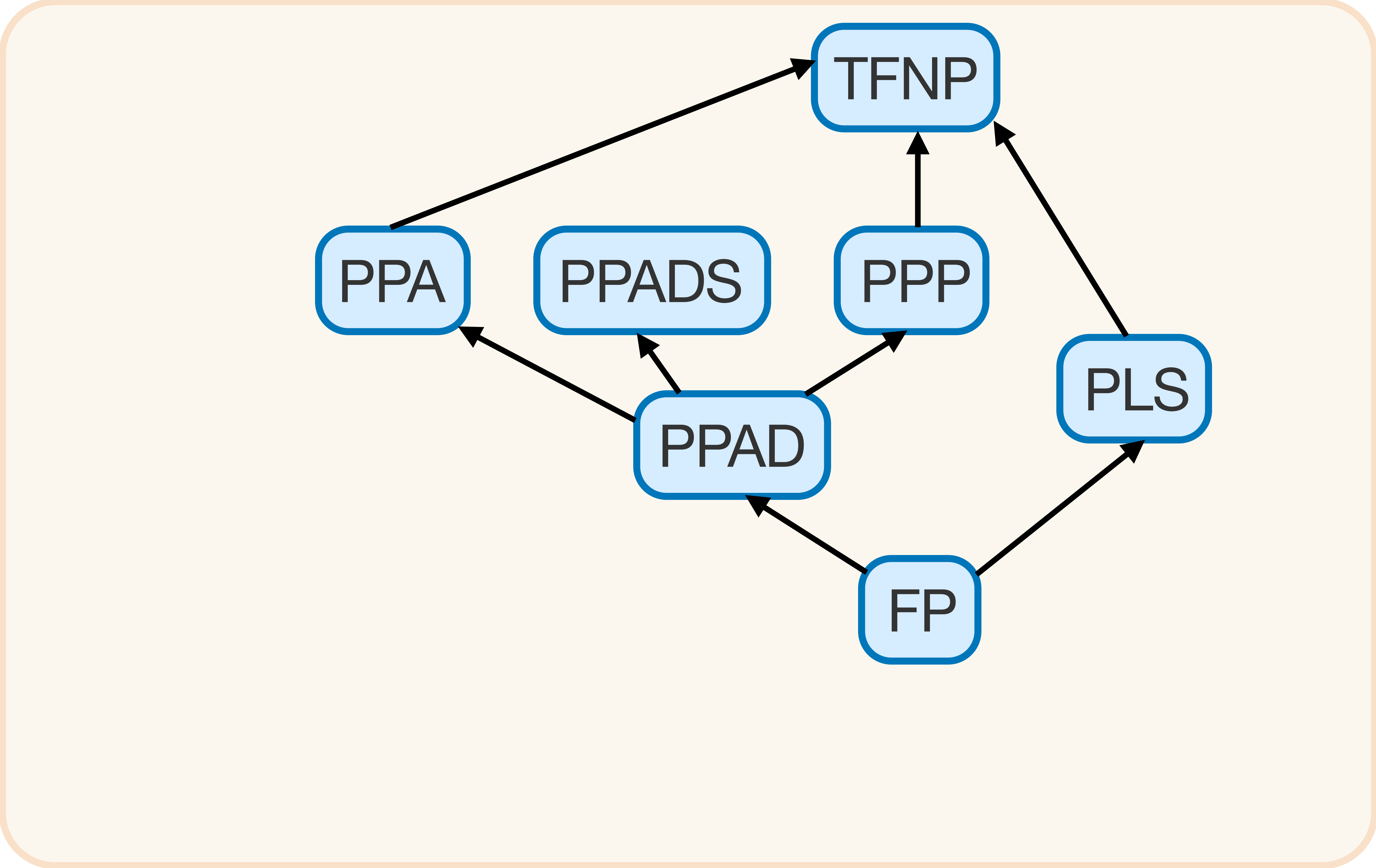
→Organizes them into a variety of classes with complete problems



Typically study the Turing Machine complexity of total search problems

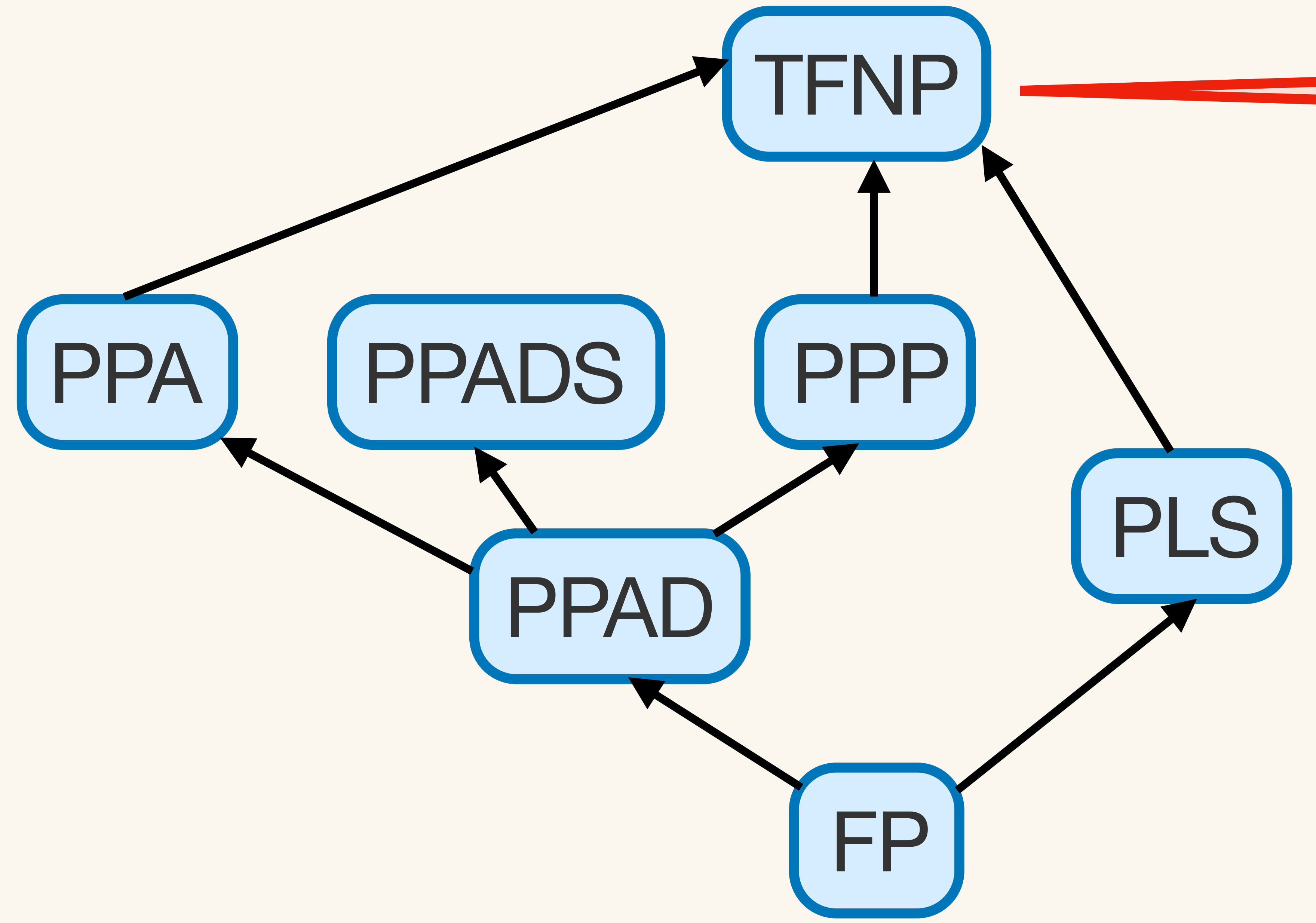
However, useful to consider other models of computation

TFNP



Model of Computation: Decision Trees

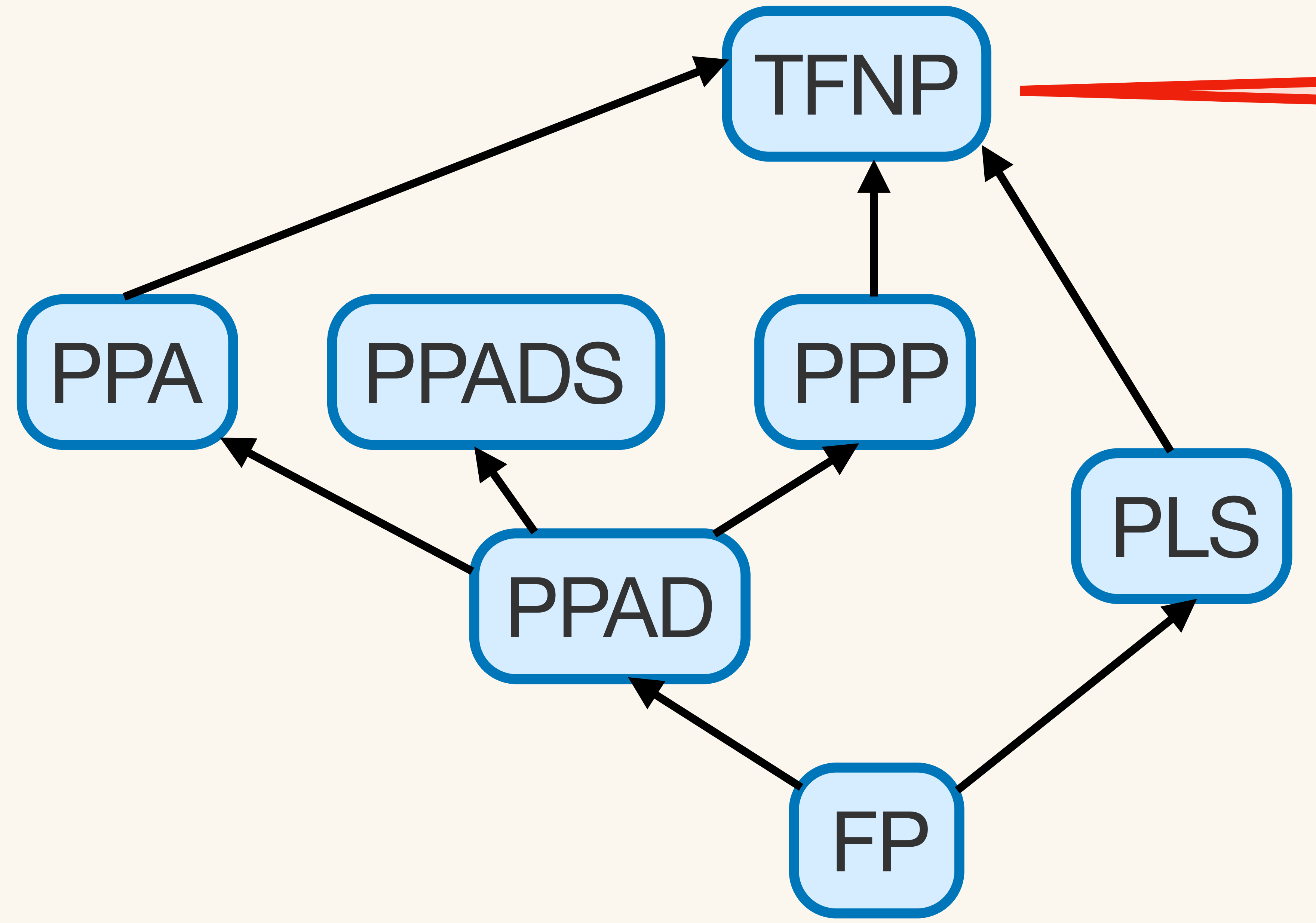
TFNP



$S \subseteq \{0,1\}^n \times \mathcal{O}$ is in $TFNP^{dt}$ if solutions can be verified by low-depth decision trees

Model of Computation: Decision Trees

TFNP



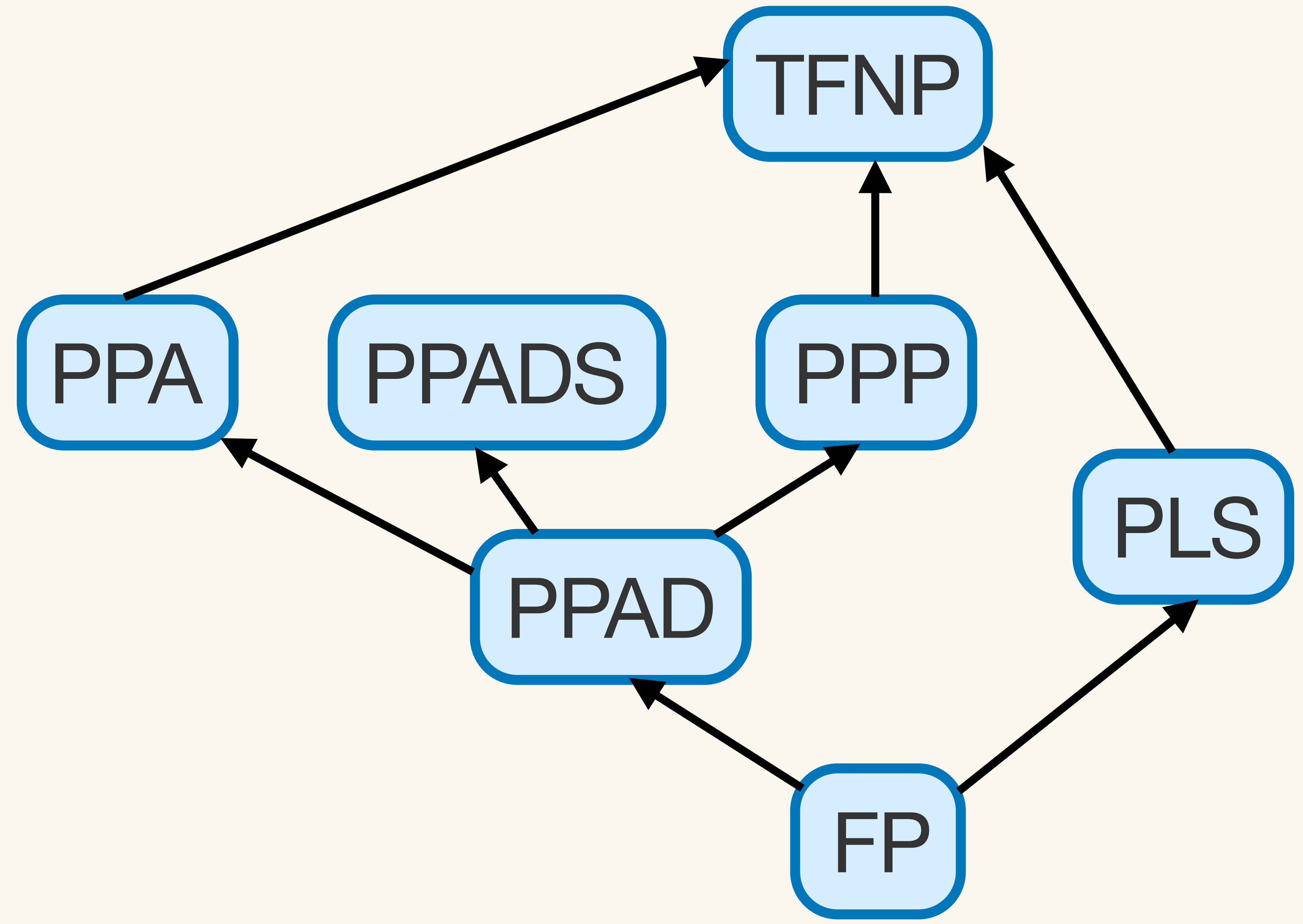
$S \subseteq \{0,1\}^n \times \mathcal{O}$ is in $TFNP^{dt}$ if solutions can be verified by low-depth decision trees

$\forall \ell \in \mathcal{O}$ there is $polylog(n)$ -depth T_ℓ such that $(x, \ell) \in S \iff T_\ell(x) = 1$

Model of Computation: Decision Trees

TFNP

[BCEIP98] Separations imply black-box / generic oracle separations

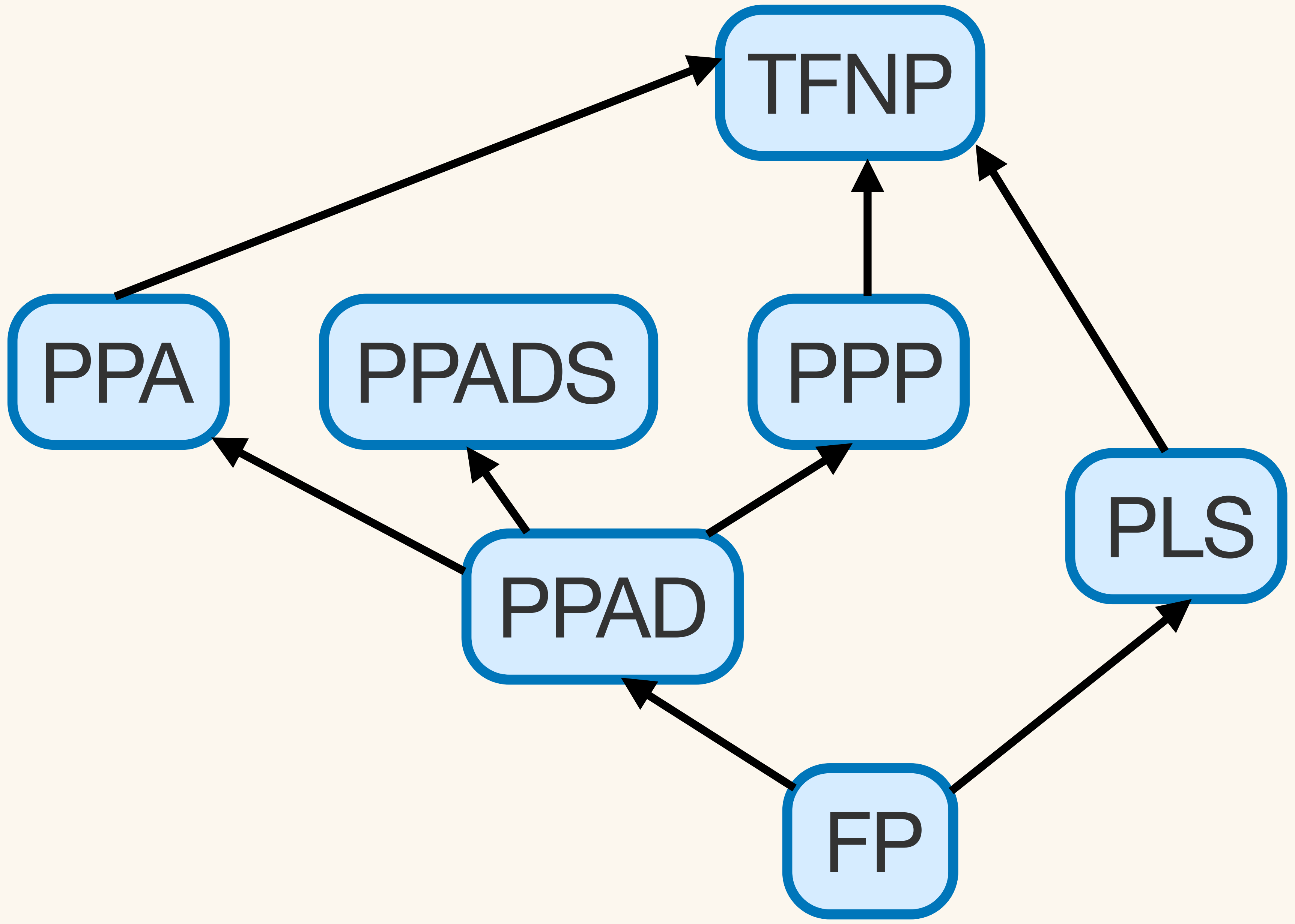


Model of Computation: Decision Trees

TFNP

[BCEIP98] Separations imply black-box / generic oracle separations

[GKRS18] Certain proof systems are **equivalent** to decision tree TFNP classes!



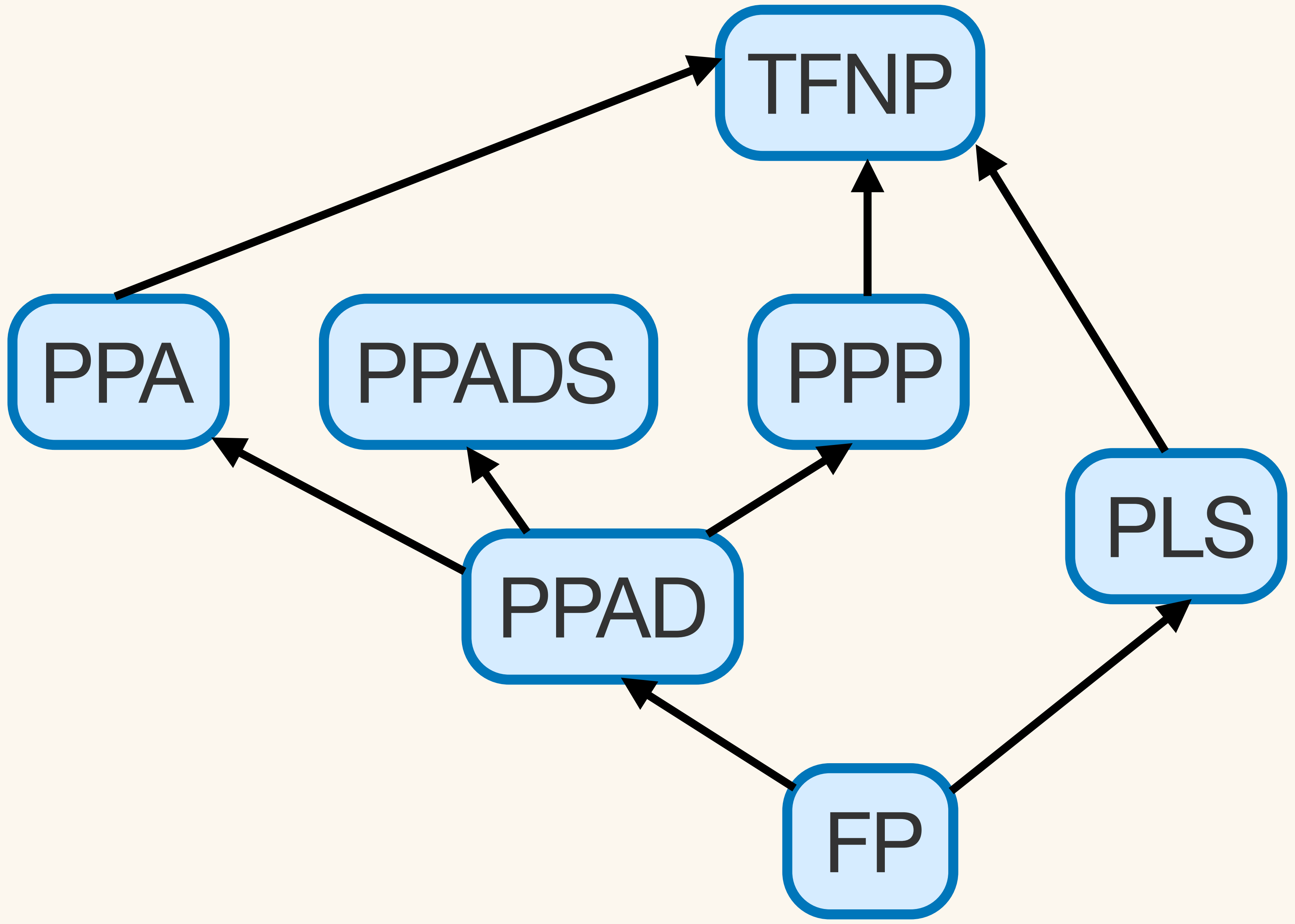
Model of Computation: Decision Trees

TFNP

[BCEIP98] Separations imply black-box / generic oracle separations

[GKRS18] Certain proof systems are **equivalent** to decision tree TFNP classes!

Say that these proof systems are characterized by the TFNP class



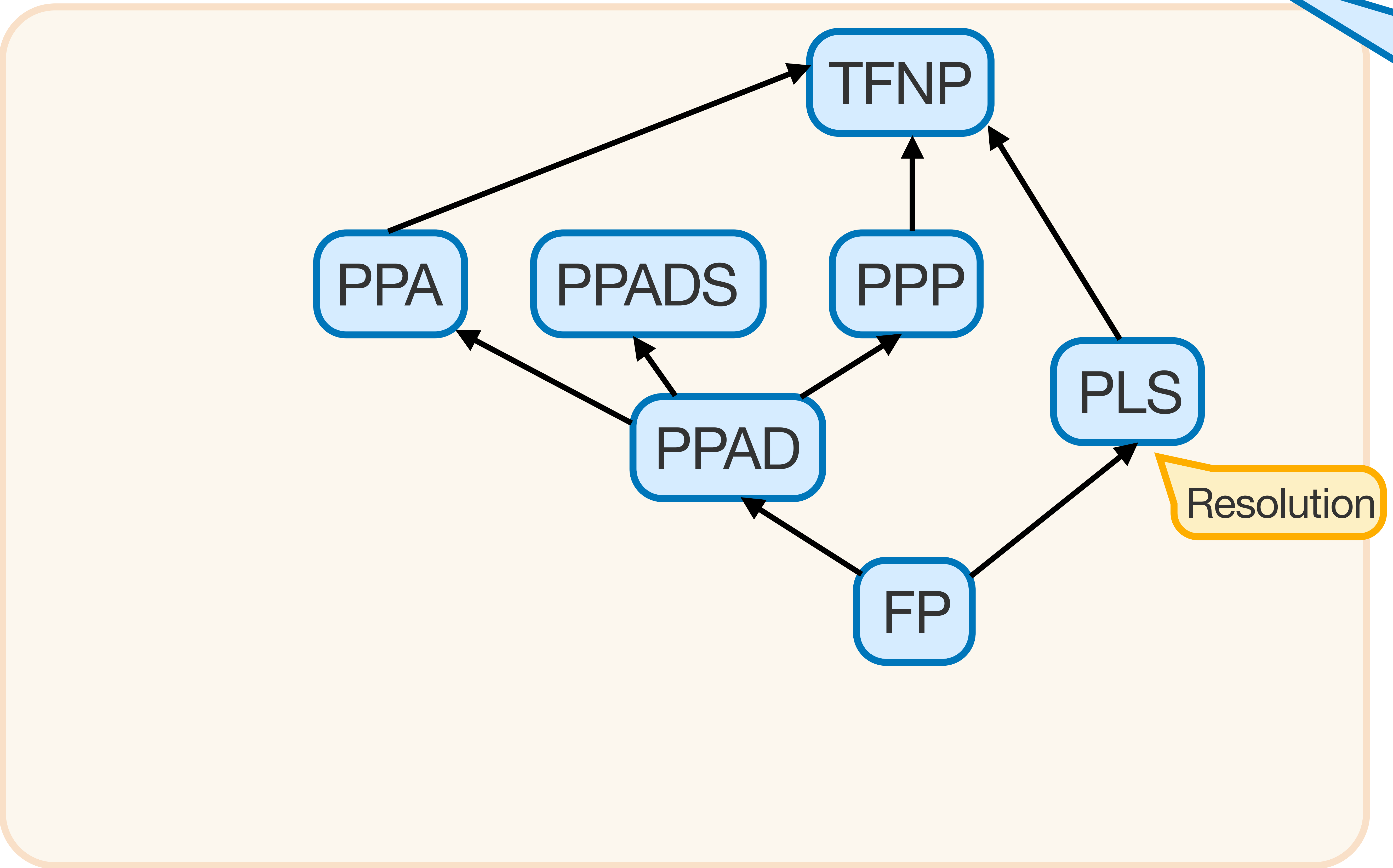
Model of Computation: Decision Trees

TFNP

[BCEIP98] Separations imply black-box / generic oracle separations

[GKRS18] Certain proof systems are **equivalent** to decision tree TFNP classes!

Say that these proof systems are characterized by the TFNP class



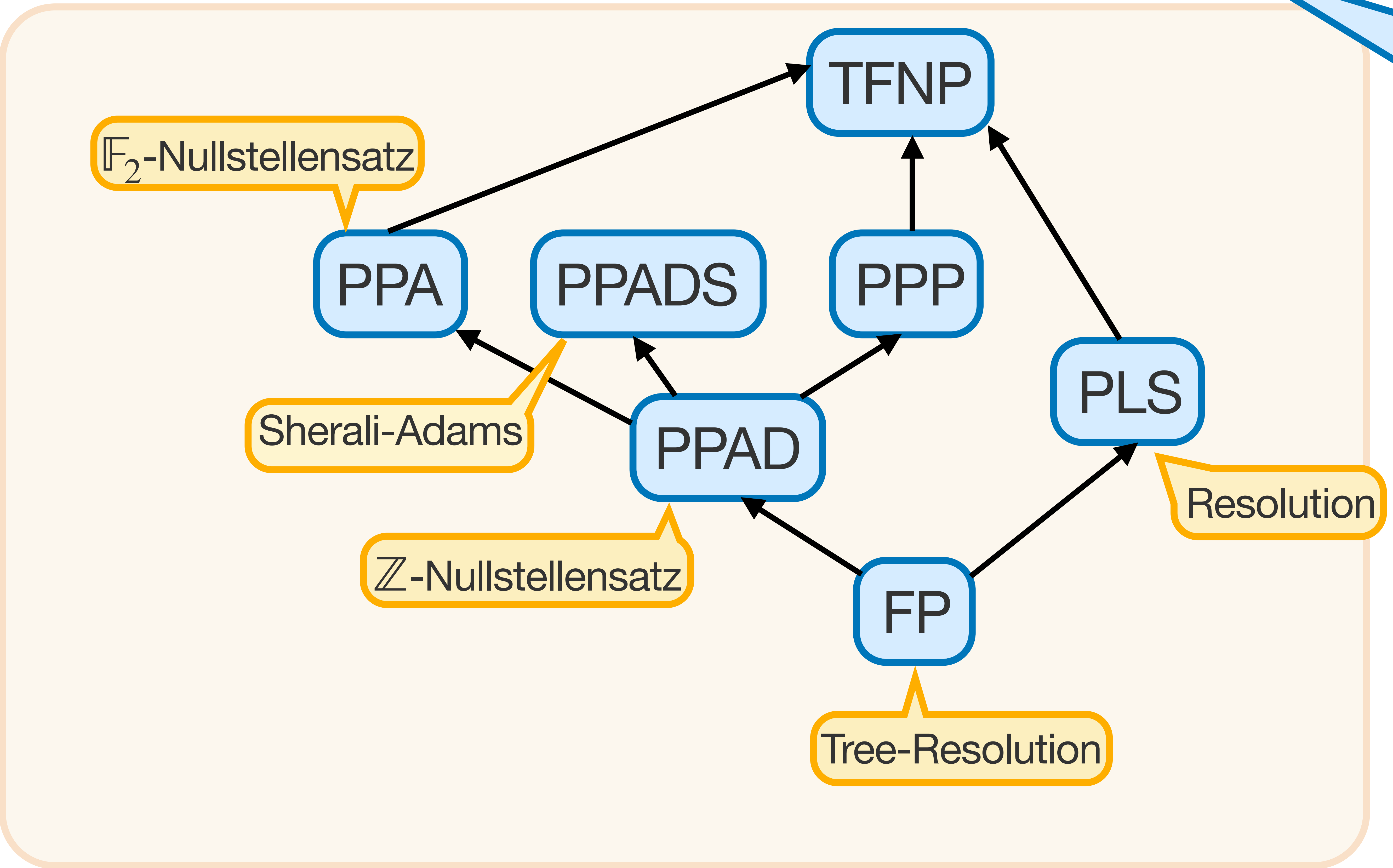
Model of Computation: Decision Trees

TFNP

[BCEIP98] Separations imply black-box / generic oracle separations

[GKRS18] Certain proof systems are **equivalent** to decision tree TFNP classes!

Say that these proof systems are characterized by the TFNP class



Model of Computation: Decision Trees

TFNP and Proof Complexity

Why? $TFNP^{dt}$ is the study of the false clause search problem!

TFNP and Proof Complexity

Why? $TFNP^{dt}$ is the study of the false clause search problem!

Claim: Any $R \subseteq \{0,1\}^n \times \mathcal{O}$ with $R \in TFNP^{dt}$ is equivalent to $Search_F$ for some unsatisfiable CNF F

TFNP and Proof Complexity

Why? $TFNP^{dt}$ is the study of the false clause search problem!

Claim: Any $R \subseteq \{0,1\}^n \times \mathcal{O}$ with $R \in TFNP^{dt}$ is equivalent to $Search_F$ for some unsatisfiable CNF F

As $R \in TFNP^{dt}$ there are $\{T_\ell\}$

TFNP and Proof Complexity

Why? $TFNP^{dt}$ is the study of the false clause search problem!

Claim: Any $R \subseteq \{0,1\}^n \times \mathcal{O}$ with $R \in TFNP^{dt}$ is equivalent to $Search_F$ for some unsatisfiable CNF F

As $R \in TFNP^{dt}$ there are $\{T_\ell\}$

Let $DNF(T_\ell)$ be obtained by taking disjunction over all 1-paths in T_ℓ

TFNP and Proof Complexity

Why? $TFNP^{dt}$ is the study of the false clause search problem!

Claim: Any $R \subseteq \{0,1\}^n \times \mathcal{O}$ with $R \in TFNP^{dt}$ is equivalent to $Search_F$ for some unsatisfiable CNF F

As $R \in TFNP^{dt}$ there are $\{T_\ell\}$

Let $DNF(T_\ell)$ be obtained by taking disjunction over all 1-paths in T_ℓ

$$F = \bigwedge_{\ell \in \mathcal{O}} \neg DNF(T_\ell)$$

TFNP and Proof Complexity

Why? $TFNP^{dt}$ is the study of the false clause search problem!

Claim: Any $R \subseteq \{0,1\}^n \times \mathcal{O}$ with $R \in TFNP^{dt}$ is equivalent to $Search_F$ for some unsatisfiable CNF F

As $R \in TFNP^{dt}$ there are $\{T_\ell\}$

Let $DNF(T_\ell)$ be obtained by taking disjunction over all 1-paths in T_ℓ

$$F = \bigwedge_{\ell \in \mathcal{O}} \neg DNF(T_\ell)$$

Expresses that R is not total:

A clause of $\neg DNF(T_\ell)$ is false under $x \iff (x, \ell) \in R$

Resolution is PLS

TFNP subclasses defined as everything $polylog(n)$-reducible to a particular search problem

Resolution is PLS

TFNP subclasses defined as everything $polylog(n)$-reducible to a particular search problem

$S \subseteq \{0,1\}^n \times \mathcal{O}$ reduces to $R \subseteq \{0,1\}^m \times \mathcal{Q}$ if there are decision trees

S

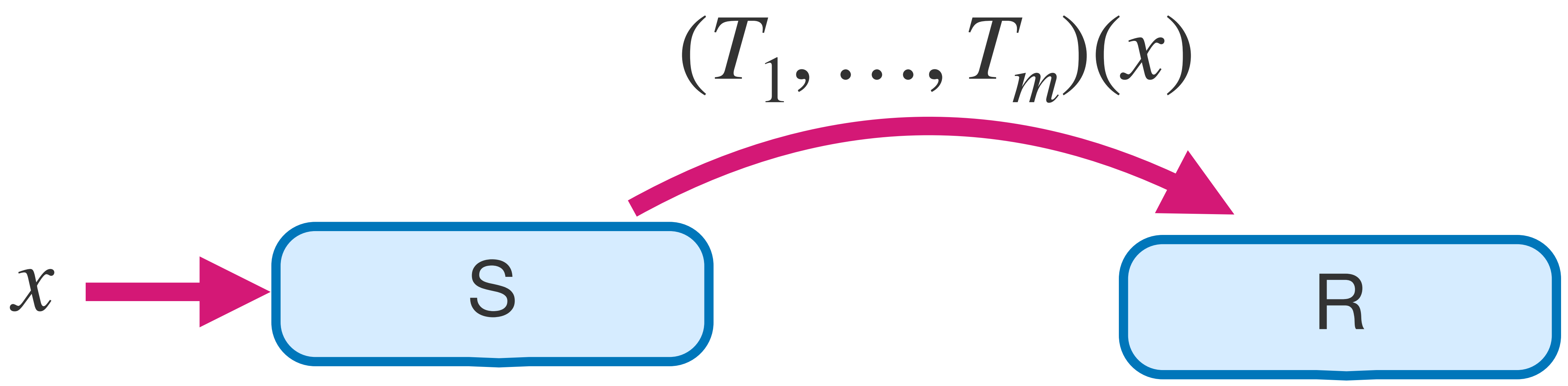
R

Resolution is PLS

TFNP subclasses defined as everything $polylog(n)$-reducible to a particular search problem

$S \subseteq \{0,1\}^n \times \mathcal{O}$ reduces to $R \subseteq \{0,1\}^m \times \mathcal{Q}$ if there are decision trees

- $T_1, \ldots, T_m$ turning inputs to S into inputs to R

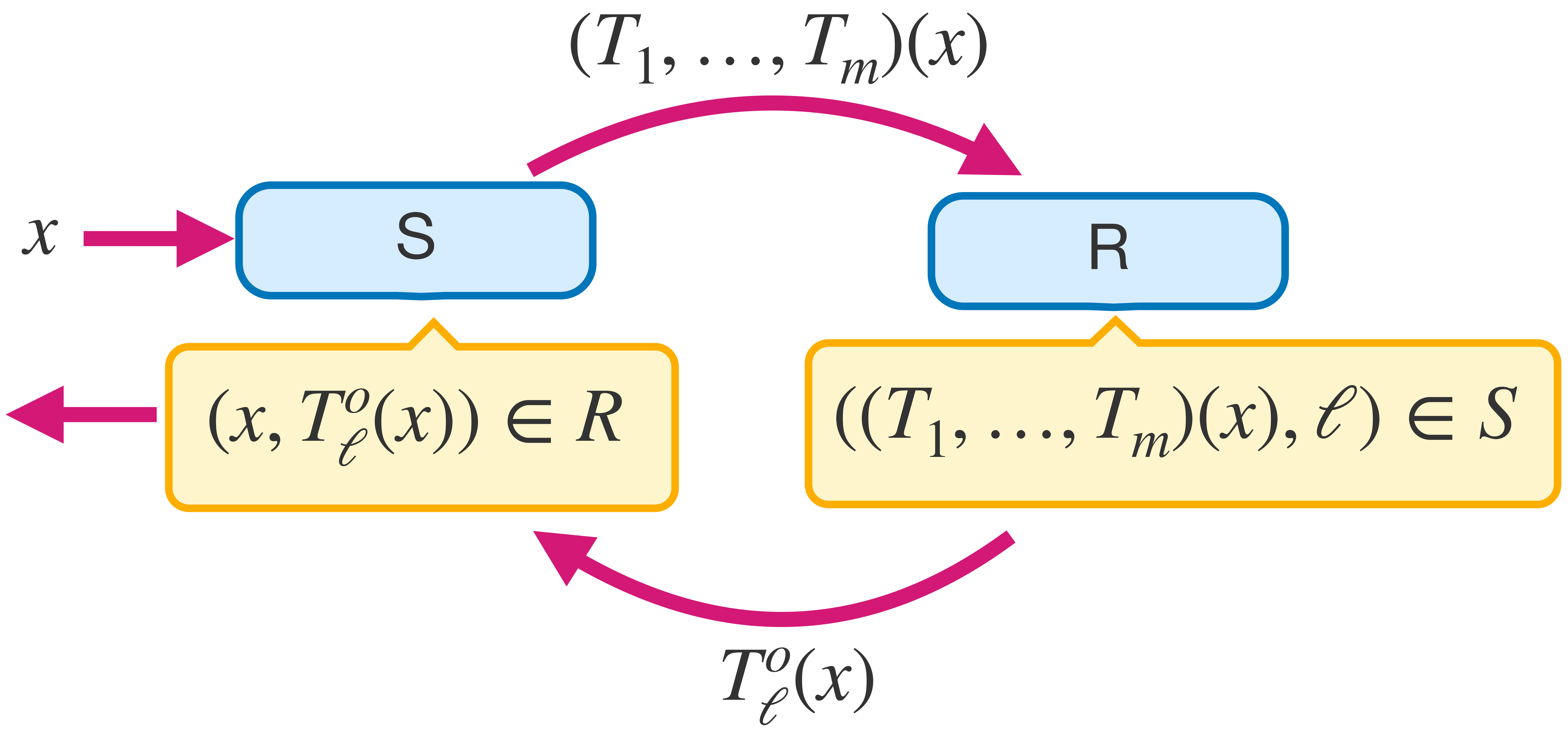


Resolution is PLS

TFNP subclasses defined as everything $polylog(n)$-reducible to a particular search problem

$S \subseteq \{0,1\}^n \times \mathcal{O}$ reduces to $R \subseteq \{0,1\}^m \times \mathcal{Q}$ if there are decision trees

- $T_1, \ldots, T_m$ turning inputs to S into inputs to R
- $T_1^o, \ldots, T_{|\mathcal{Q}|}^o$ translating solutions to R into solutions to S

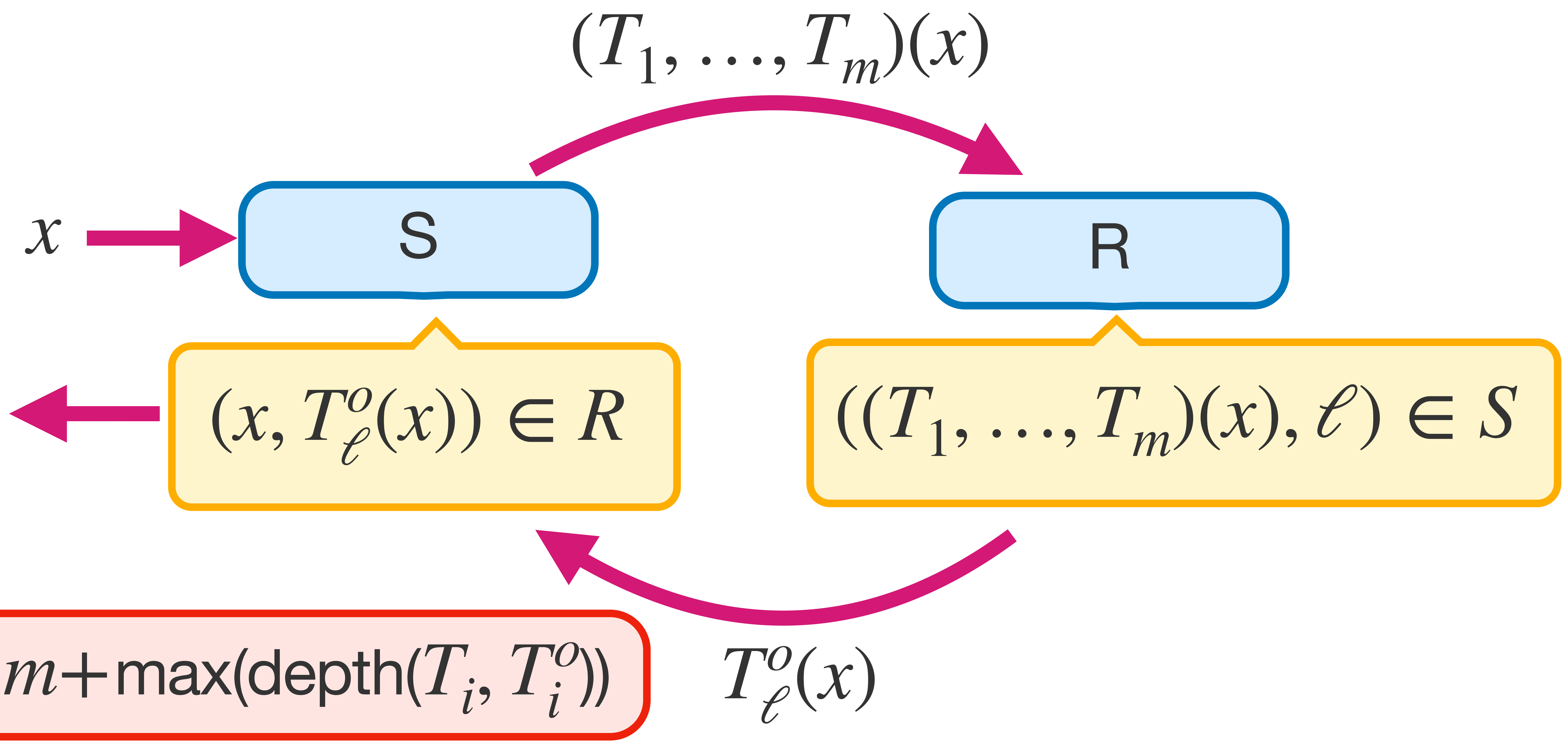


Resolution is PLS

TFNP subclasses defined as everything $polylog(n)$-reducible to a particular search problem

$S \subseteq \{0,1\}^n \times \mathcal{O}$ reduces to $R \subseteq \{0,1\}^m \times \mathcal{Q}$ if there are decision trees

- $T_1, \ldots, T_m$ turning inputs to S into inputs to R
- $T_1^o, \ldots, T_{|\mathcal{Q}|}^o$ translating solutions to R into solutions to S



Complexity: $\log m + \max(\text{depth}(T_i, T_i^o))$

Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

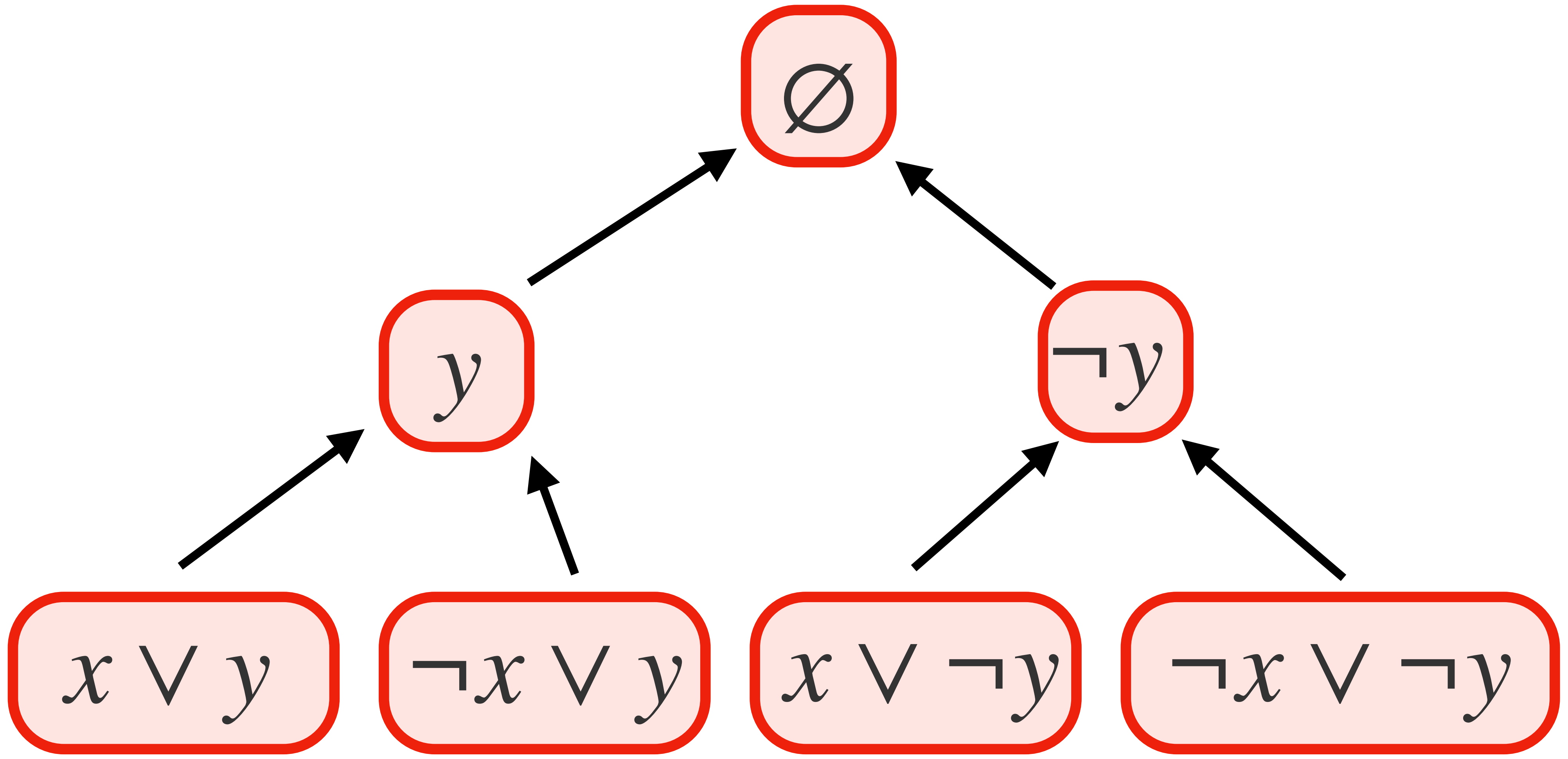
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longrightarrow$



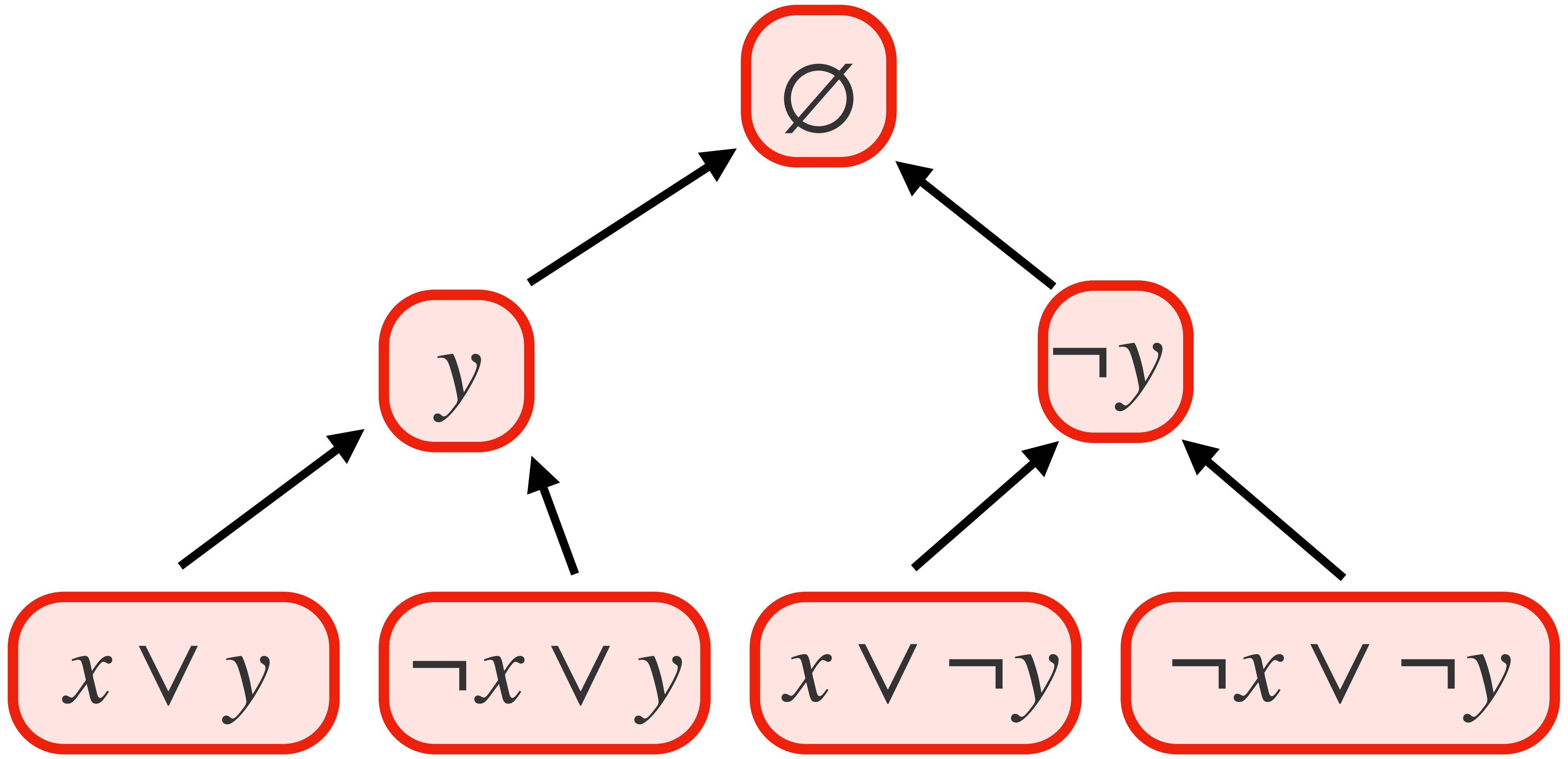
Resolution is PLS

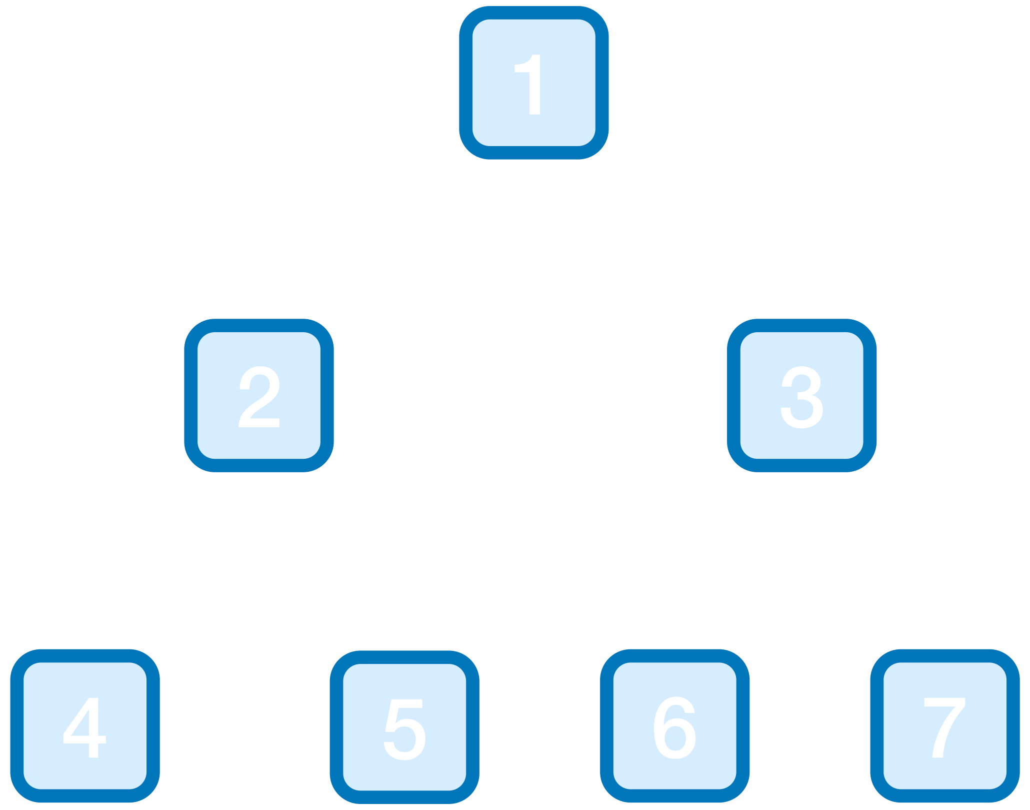
Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longrightarrow$





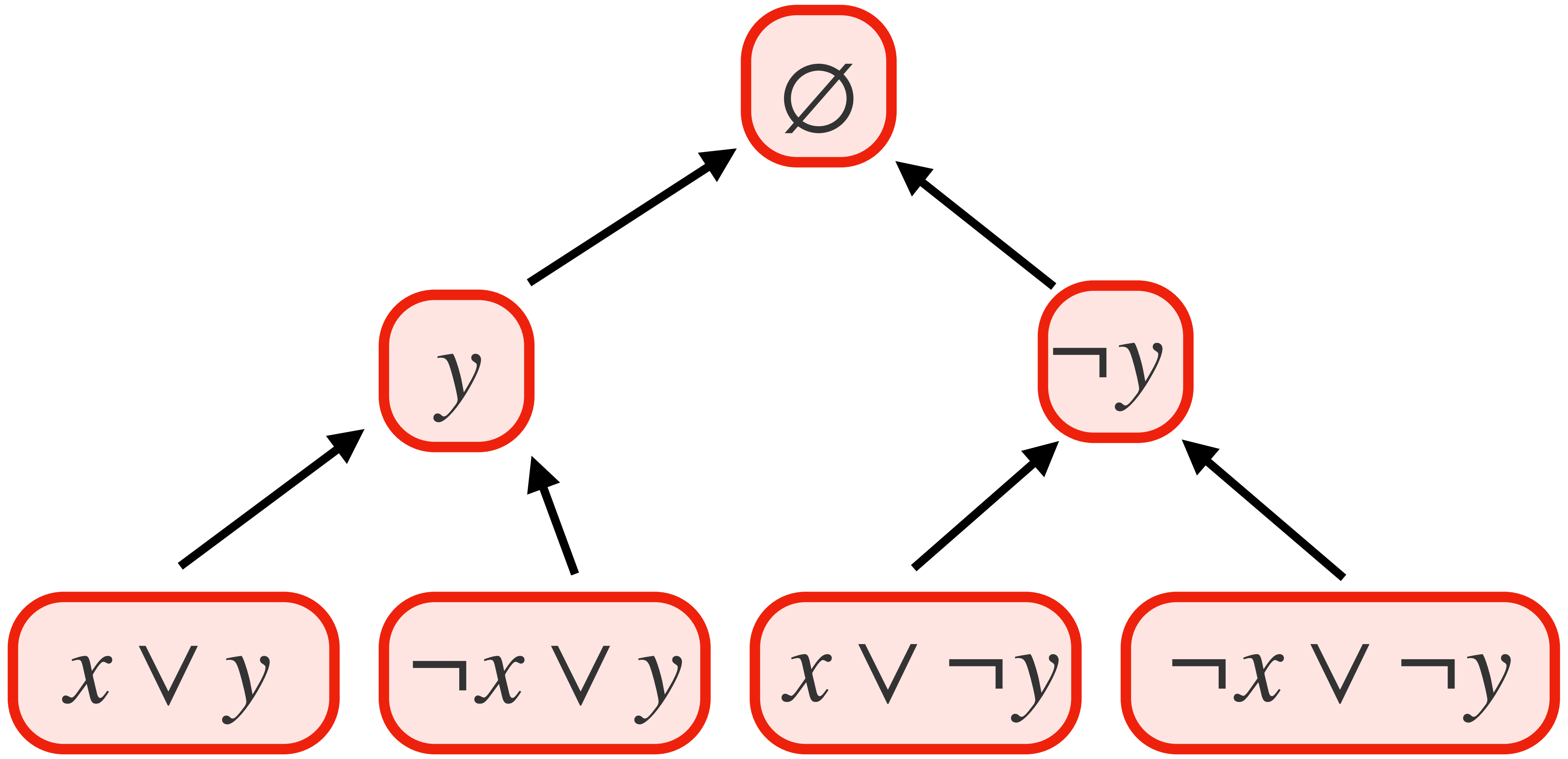
Resolution is PLS

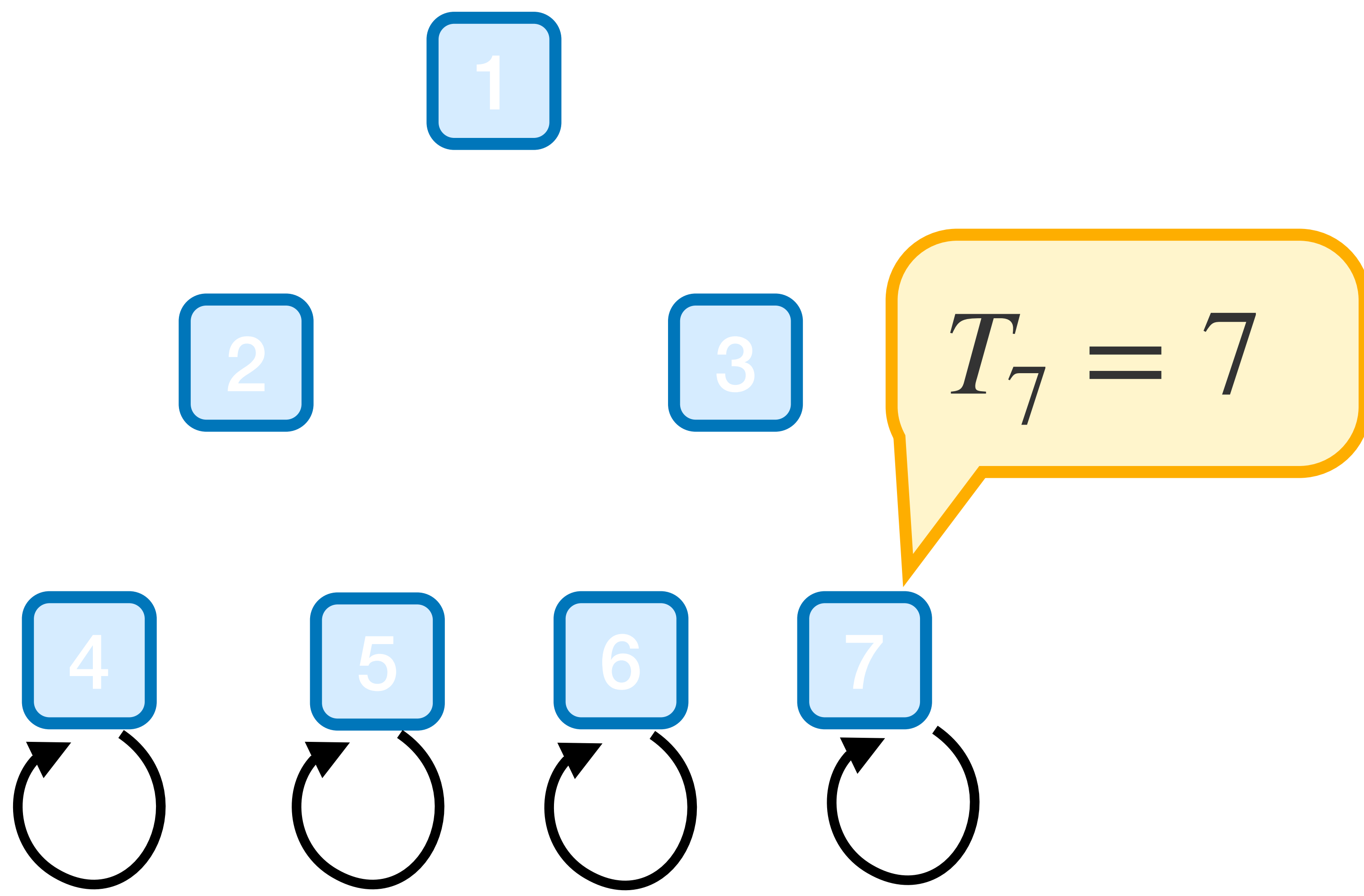
Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longrightarrow$





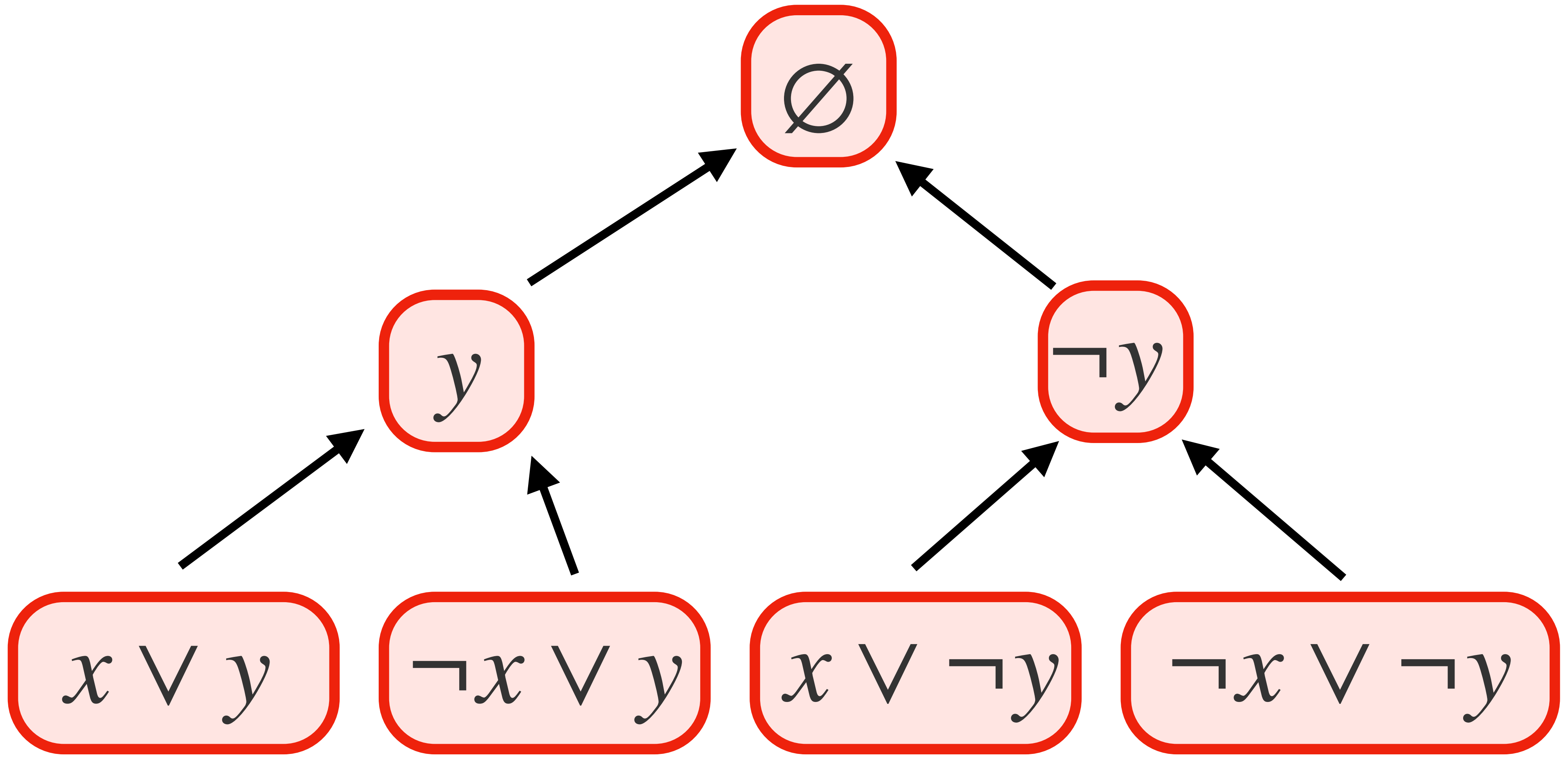
Resolution is PLS

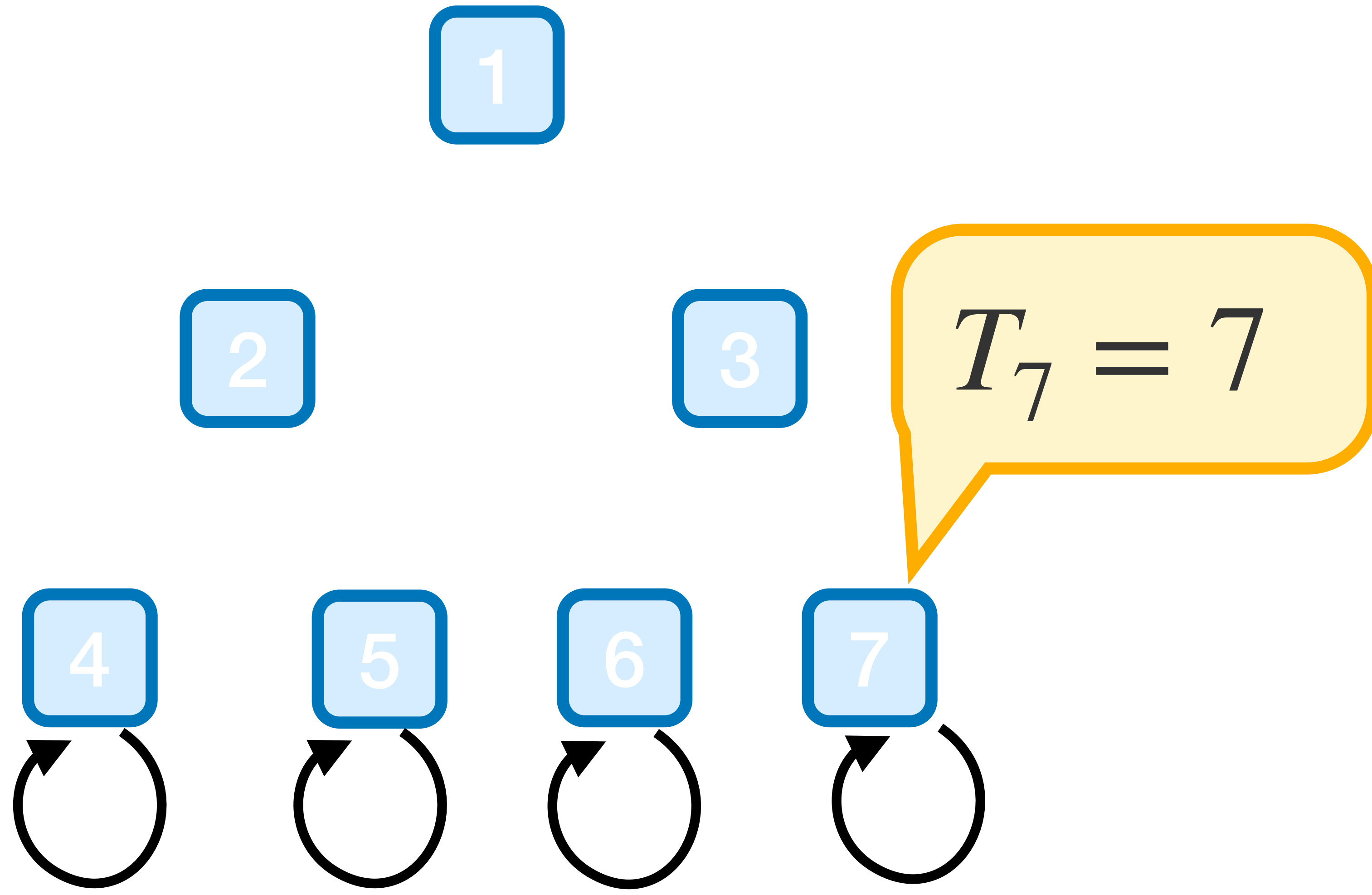
Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longrightarrow$





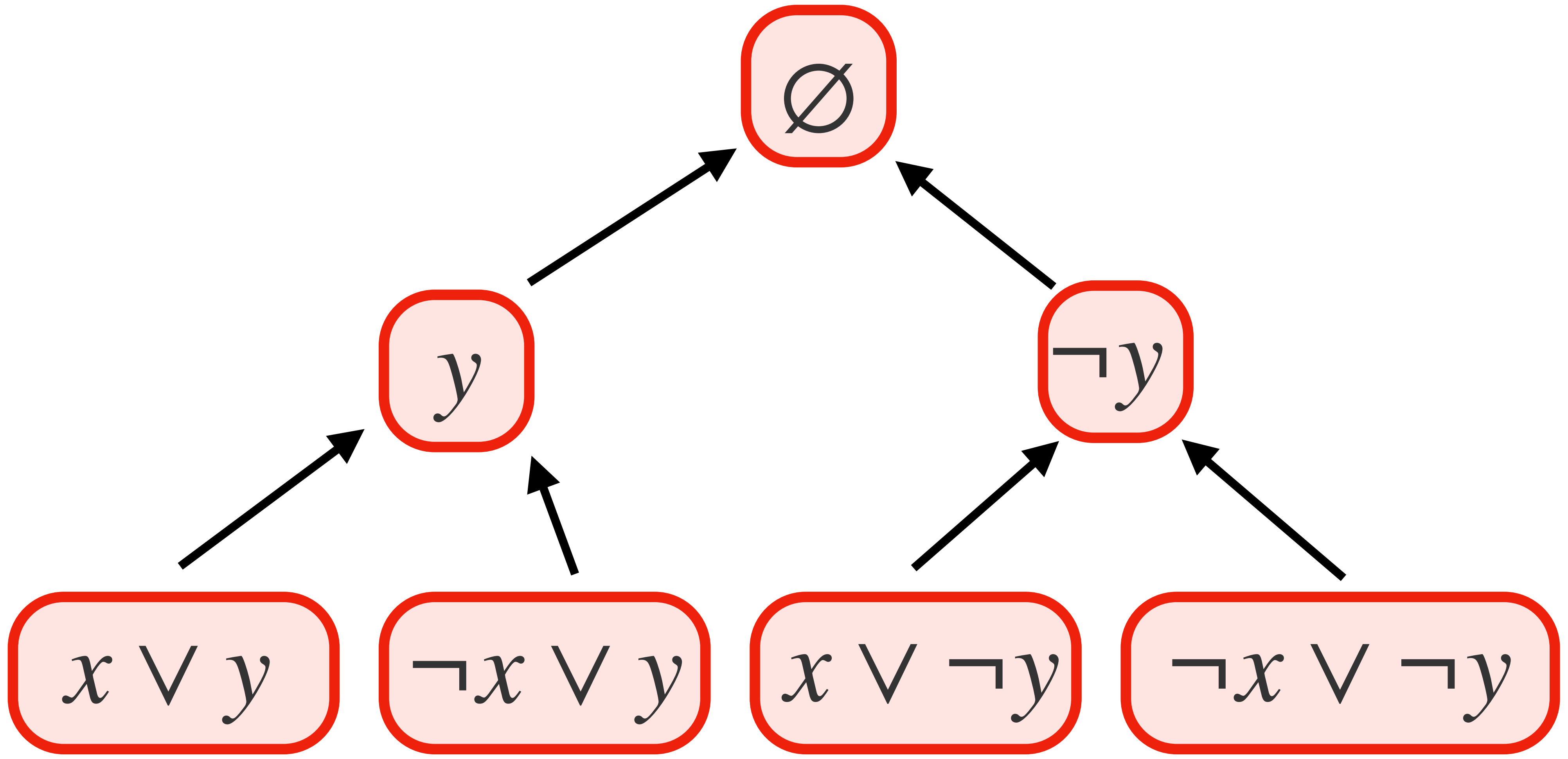
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

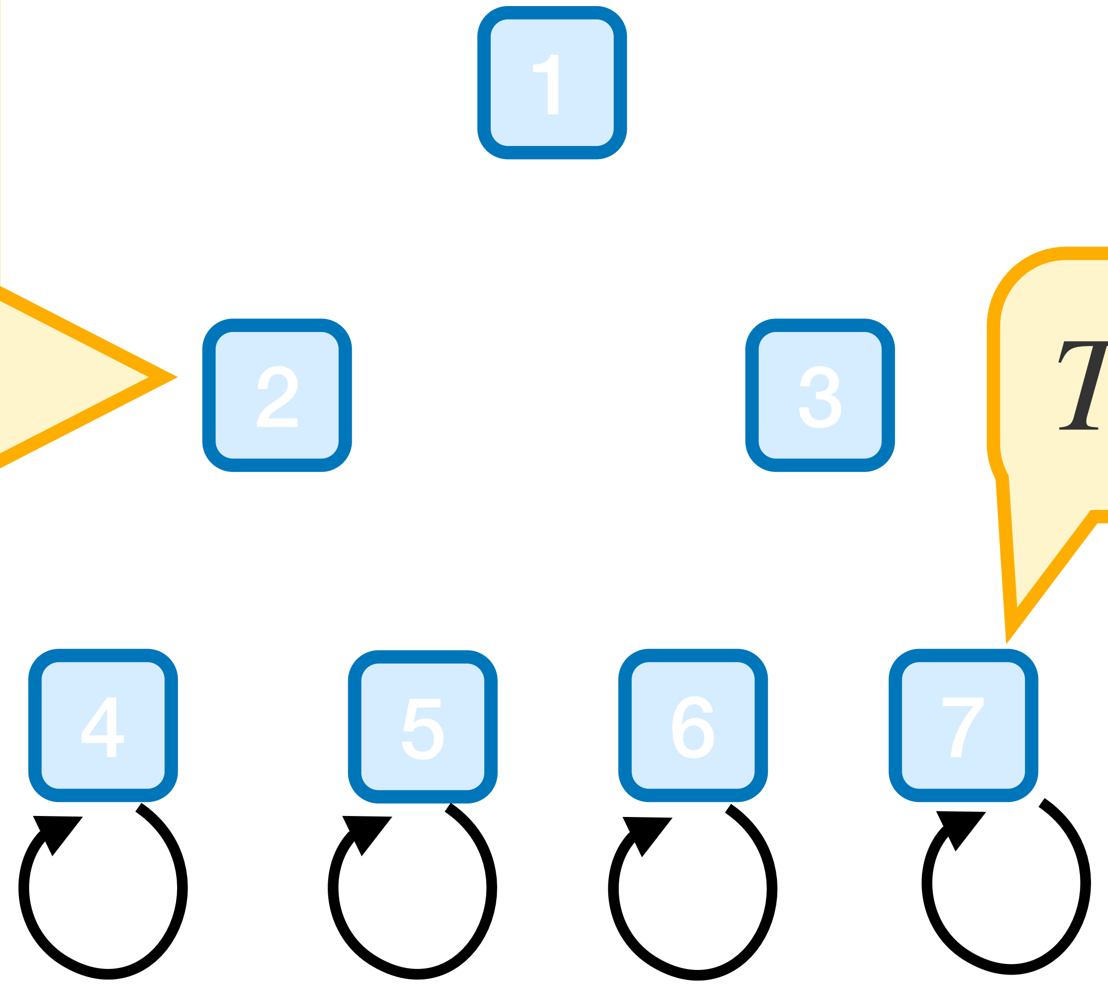
PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longrightarrow$



T_2 queries x, y:



$T_7 = 7$

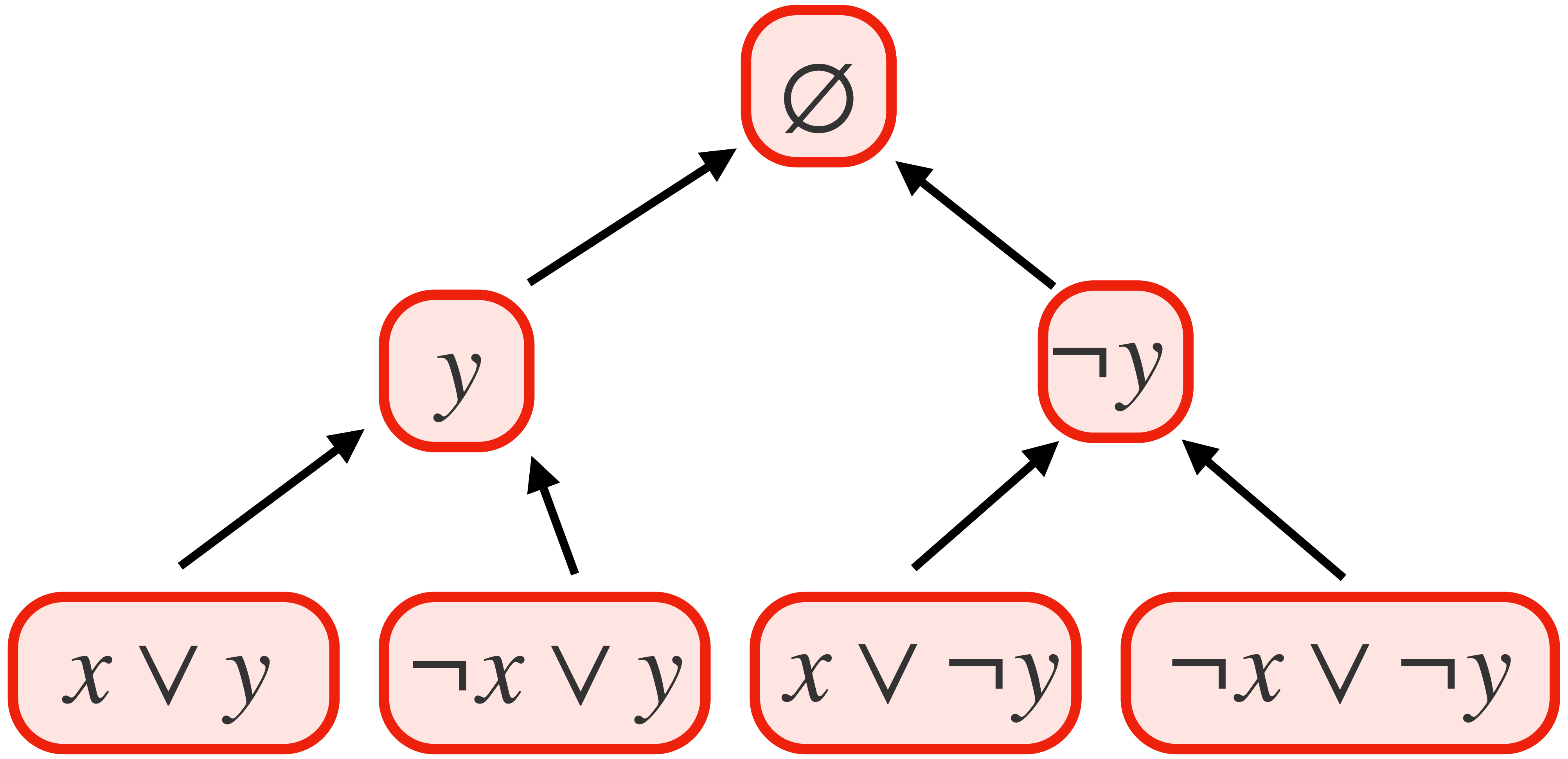
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

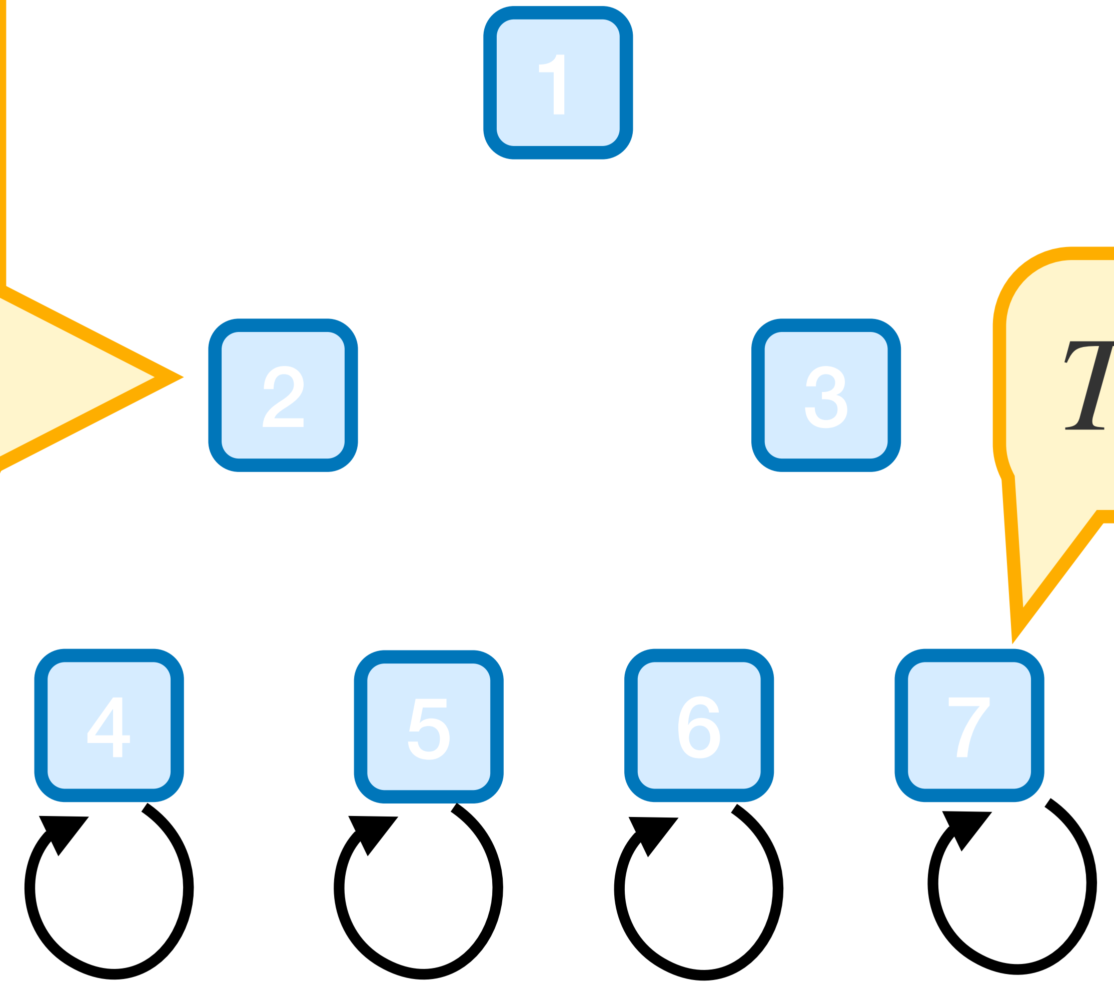
SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longrightarrow$



T_2 queries x, y:

$$T_2 = \begin{cases} 2 & \text{if } y = 1 \\ 4 & \text{if } x \vee y = 0 \\ 5 & \text{otherwise} \end{cases}$$



$T_7 = 7$

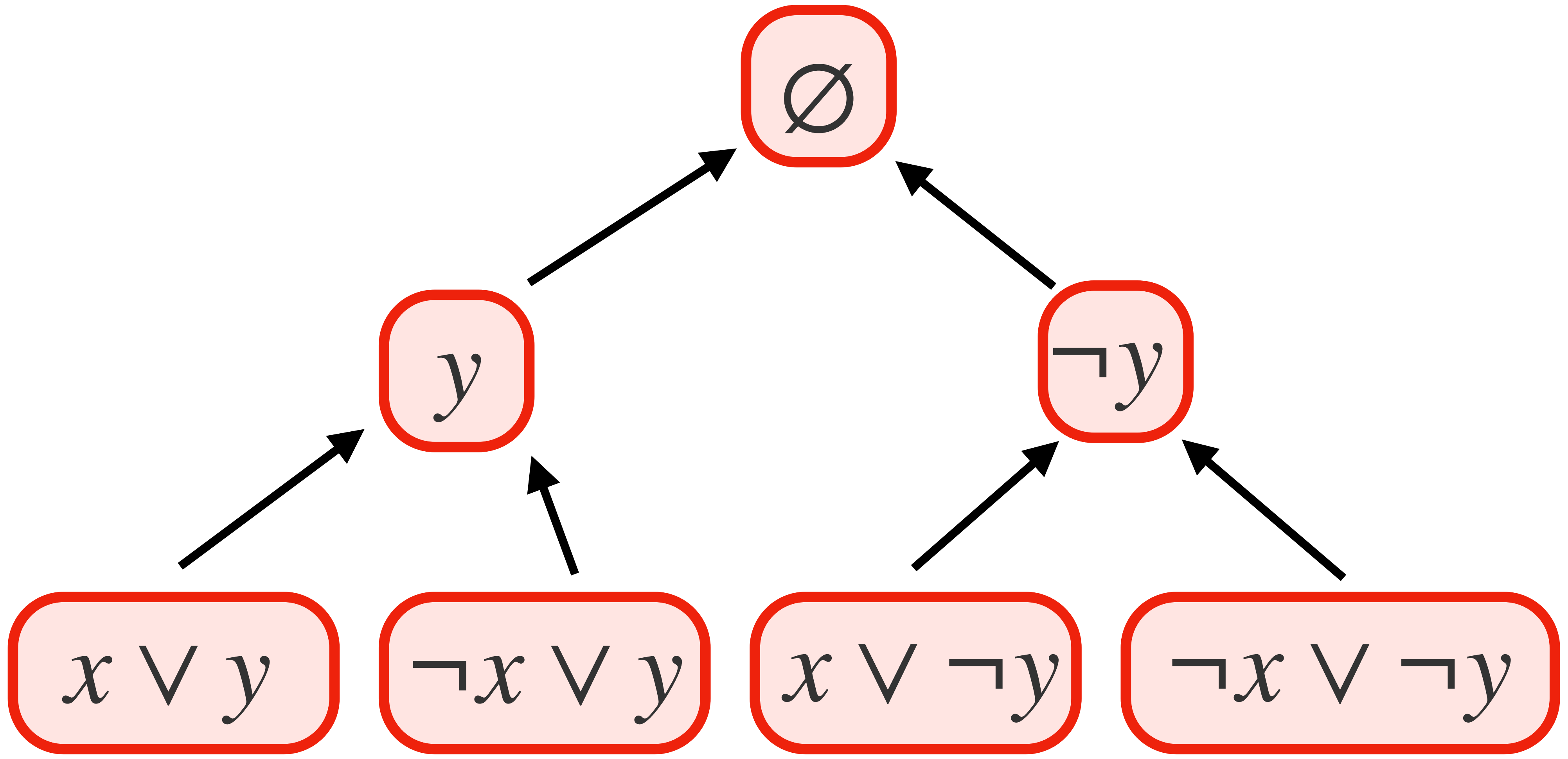
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

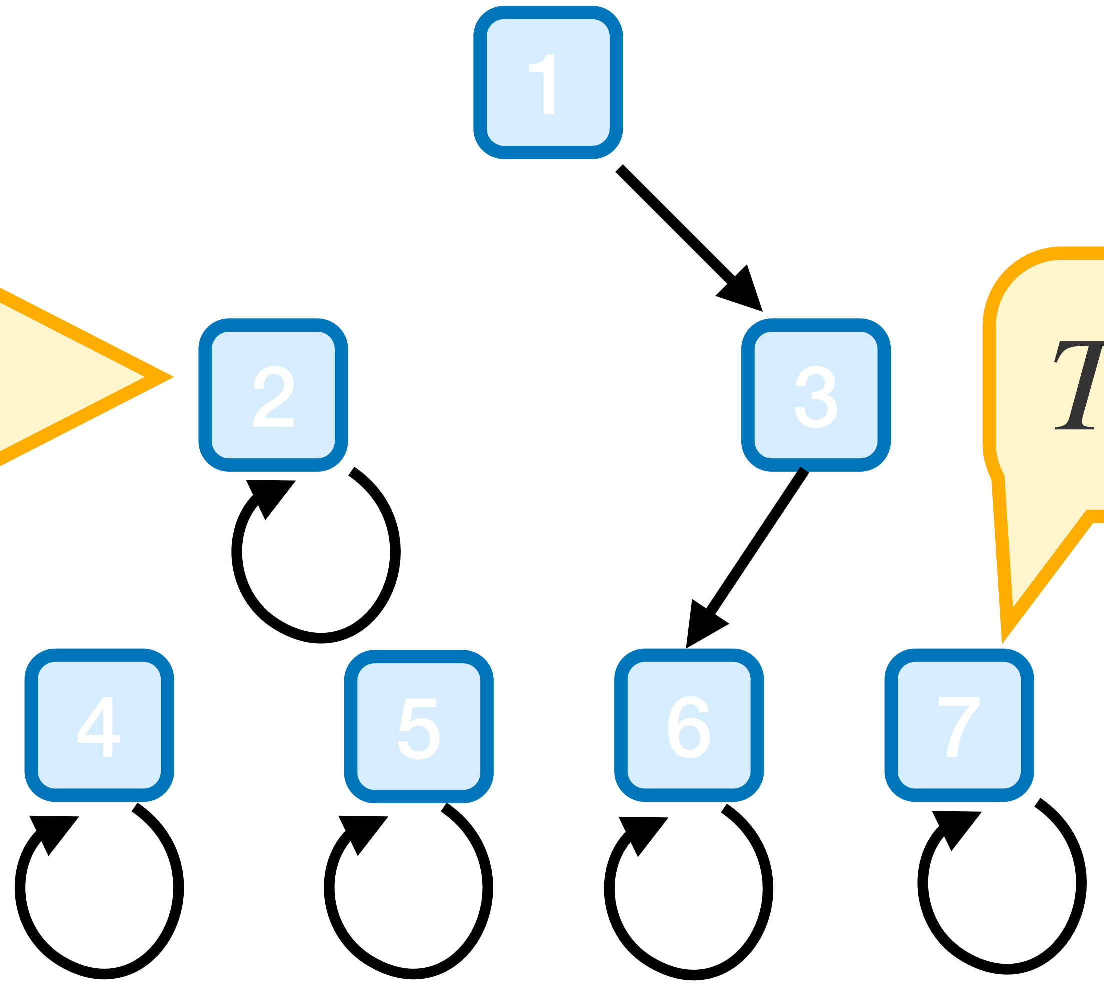
$\Longrightarrow$



T_2 queries x, y:

$$T_2 = \begin{cases} 2 & \text{if } y = 1 \\ 4 & \text{if } x \vee y = 0 \\ 5 & \text{otherwise} \end{cases}$$

e.g. $x = 01$



$T_7 = 7$

Resolution is PLS

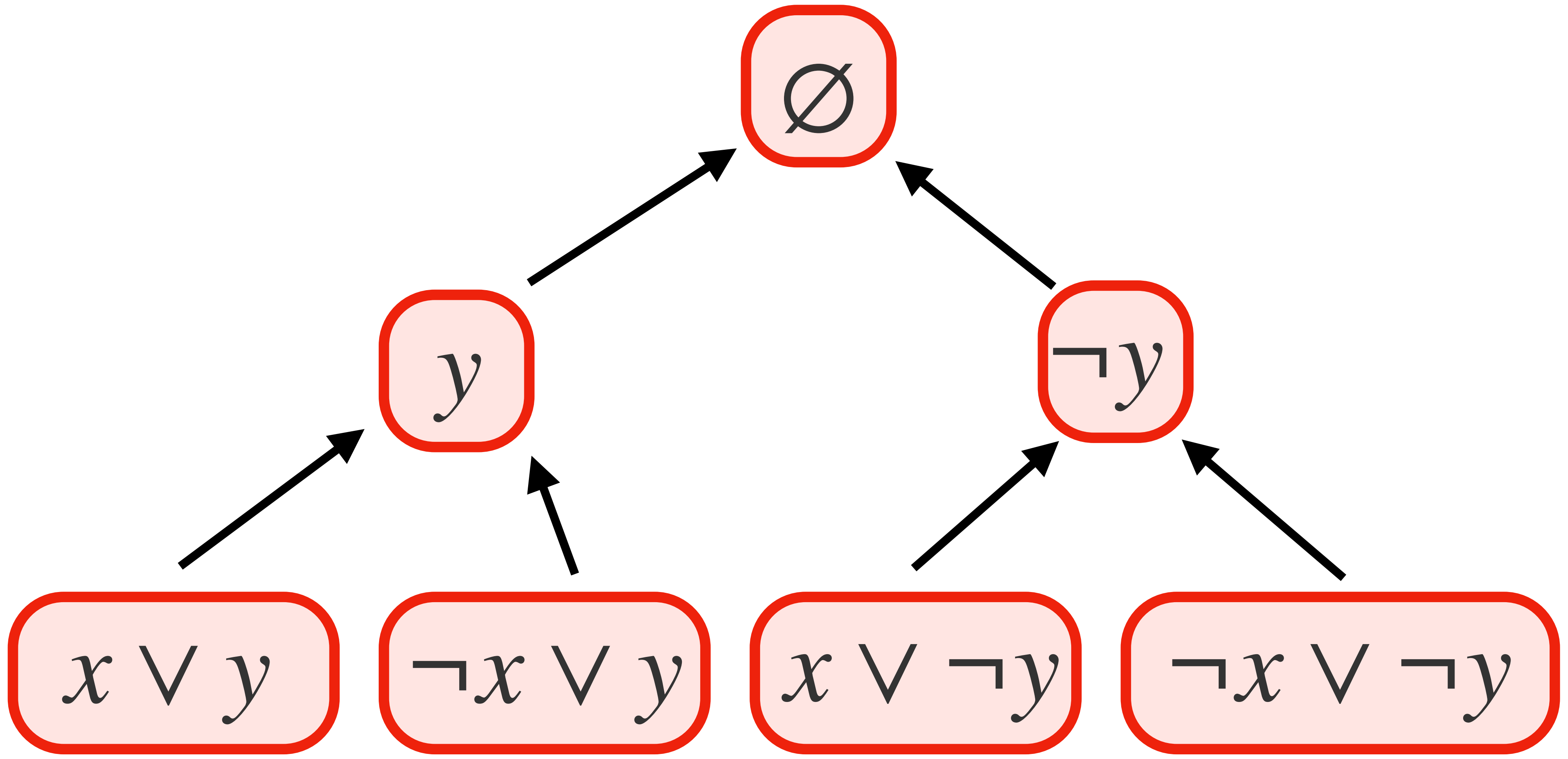
Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longrightarrow$

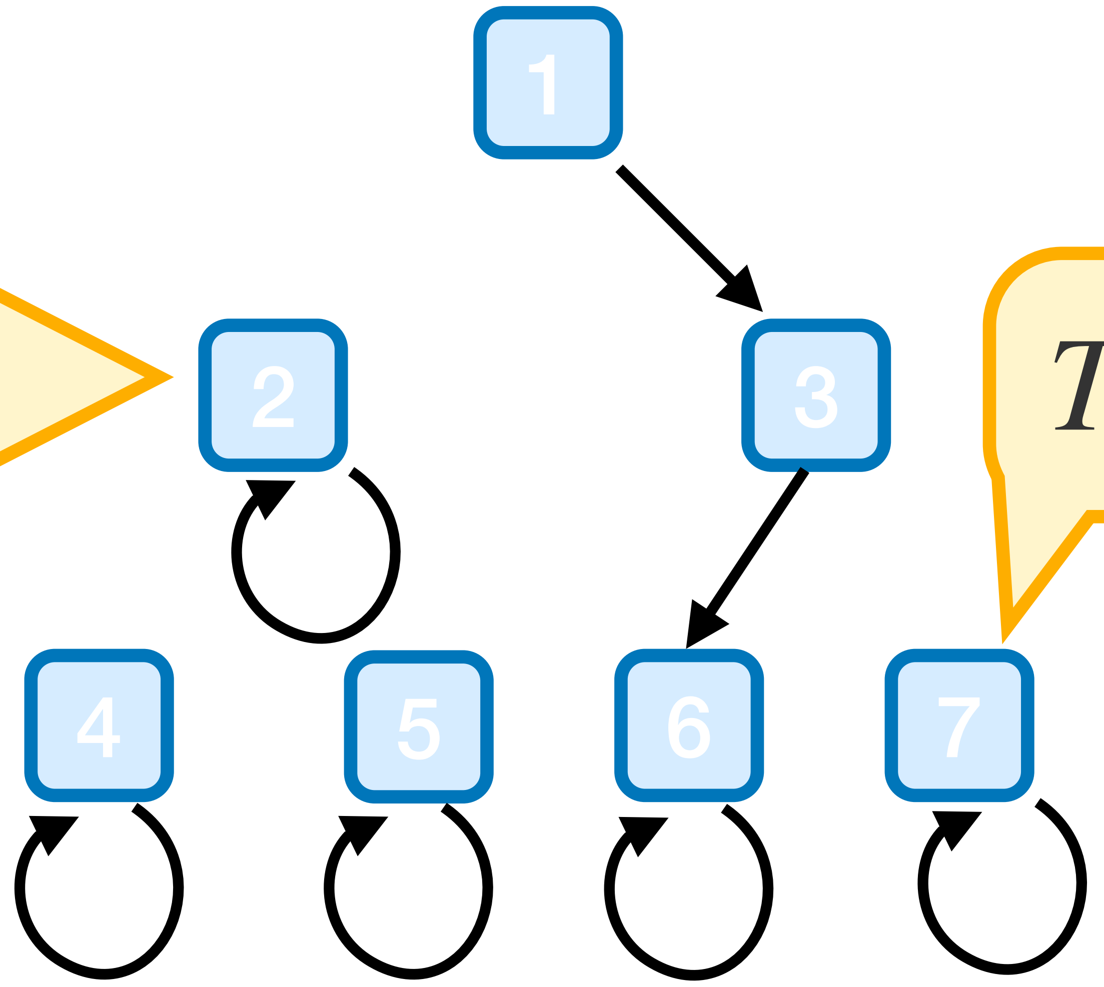
Resolution is sound $\Longrightarrow$ Solutions are false clauses!



T_2 queries x, y:

$$T_2 = \begin{cases} 2 & \text{if } y = 1 \\ 4 & \text{if } x \vee y = 0 \\ 5 & \text{otherwise} \end{cases}$$

e.g. $x = 01$



$T_7 = 7$

Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

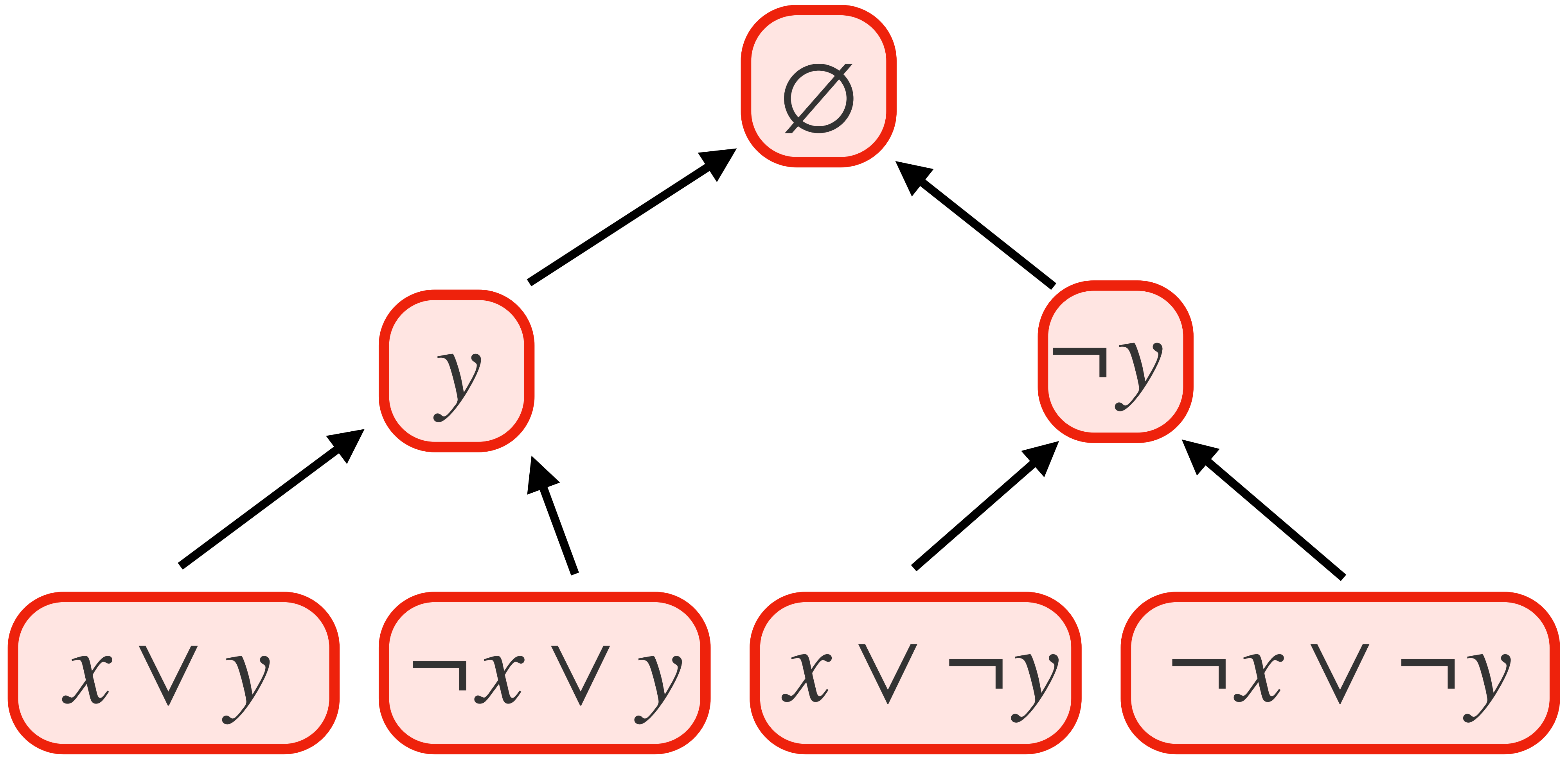
SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longrightarrow$

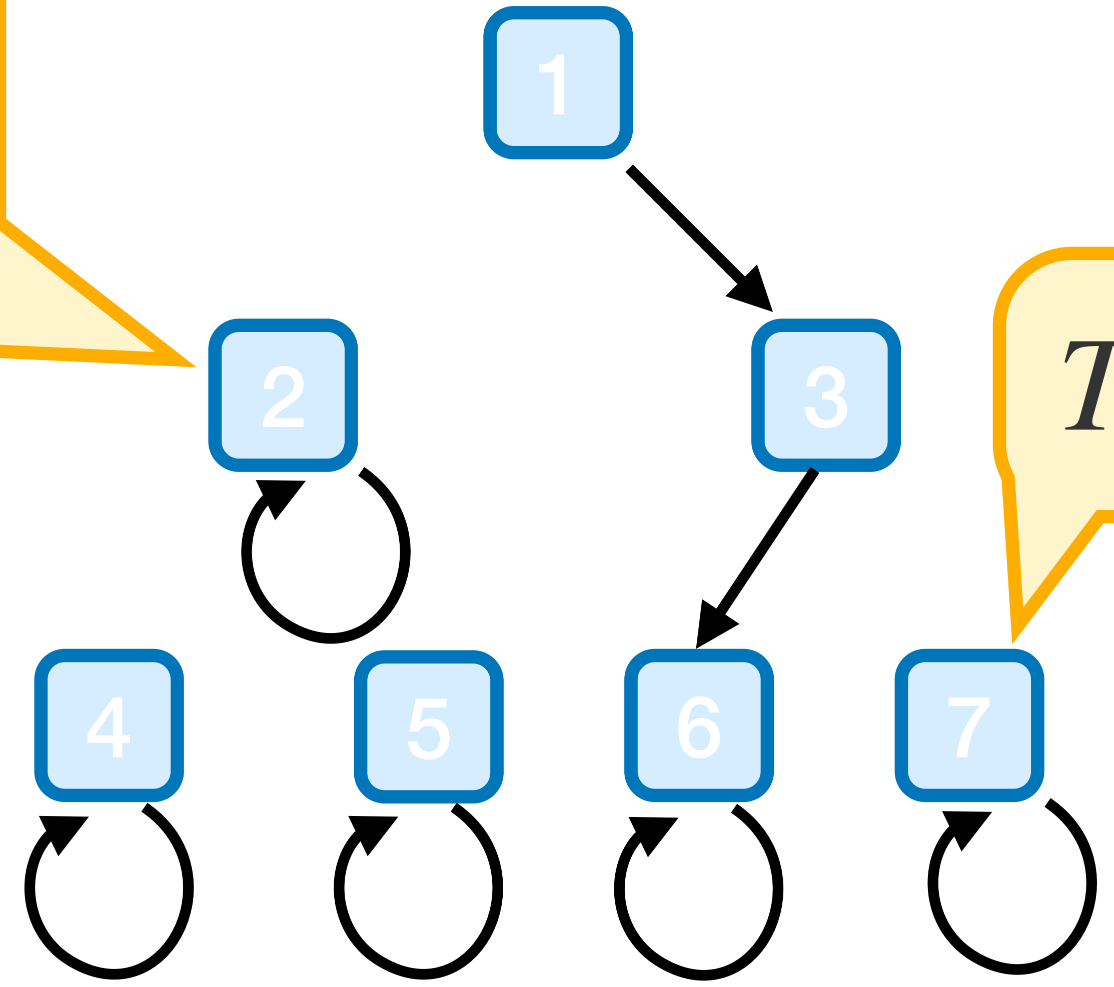
Resolution is sound $\Longrightarrow$ Solutions are false clauses!

T_2^o queries x, y:

$$T_2^o = \begin{cases} C_1 & \text{if } x \vee y = 0 \\ C_2 & \text{otherwise} \end{cases}$$



e.g. $x = 01$



$T_7 = 7$

Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$

Delayer Prover Game on F:

Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$

Delayer Prover Game on F: each round

- Query: Prover suggests a variable x_i Delayer sets $x_i \in \{0,1\}$

Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$

Delayer Prover Game on F: each round

- Query: Prover suggests a variable x_i Delayer sets $x_i \in \{0,1\}$
- Forget: Prover sets a set of variables $x_{j_1}, \ldots, x_{j_k} = *$

Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$

Delayer Prover Game on F: each round

- Query: Prover suggests a variable x_i Delayer sets $x_i \in \{0,1\}$
- Forget: Prover sets a set of variables $x_{j_1}, \ldots, x_{j_k} = *$

Game ends when current assignment falsifies a clause of F

Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Leftarrow$

Delayer Prover Game on F: each round

- Query: Prover suggests a variable x_i Delayer sets $x_i \in \{0,1\}$
- Forget: Prover sets a set of variables $x_{j_1}, \ldots, x_{j_k} = *$

Game ends when current assignment falsifies a clause of F

w-Prover Strategy: ends the game while remembering at most w variables at any time

Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1,\ldots,n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$

Delayer Prover Game on F: each round

- Query: Prover suggests a variable x_i Delayer sets $x_i \in \{0,1\}$
- Forget: Prover sets a set of variables $x_{j_1}, \ldots, x_{j_k} = *$

Game ends when current assignment falsifies a clause of F

w-Prover Strategy: ends the game while remembering at most w variables at any time

w-Prover strategy $\Longrightarrow$ Complexity $w \log n$ Resolution proof

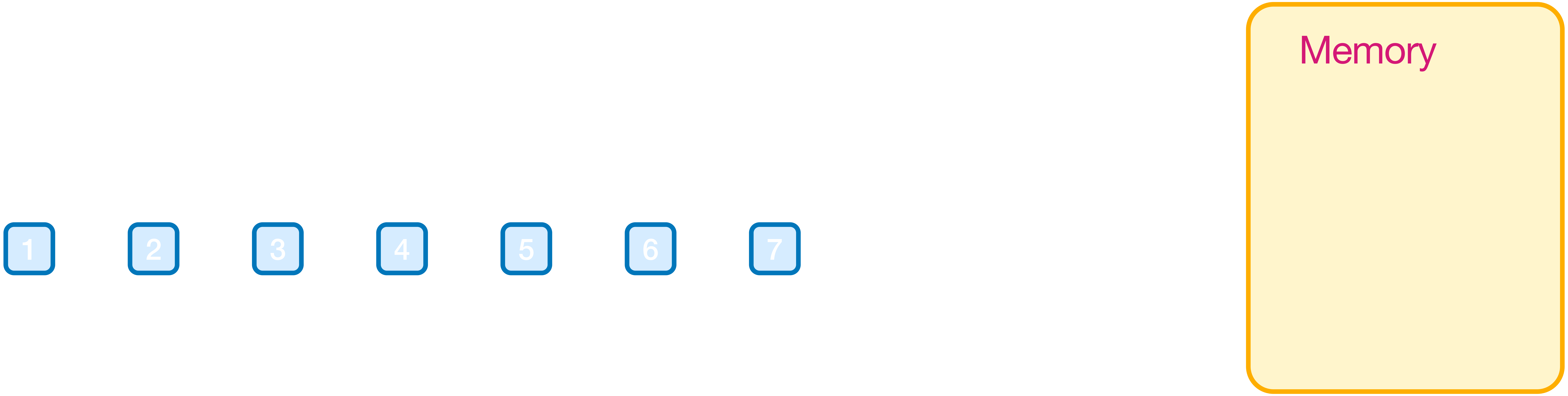
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$ Extract a Prover Strategy for $Search_F$



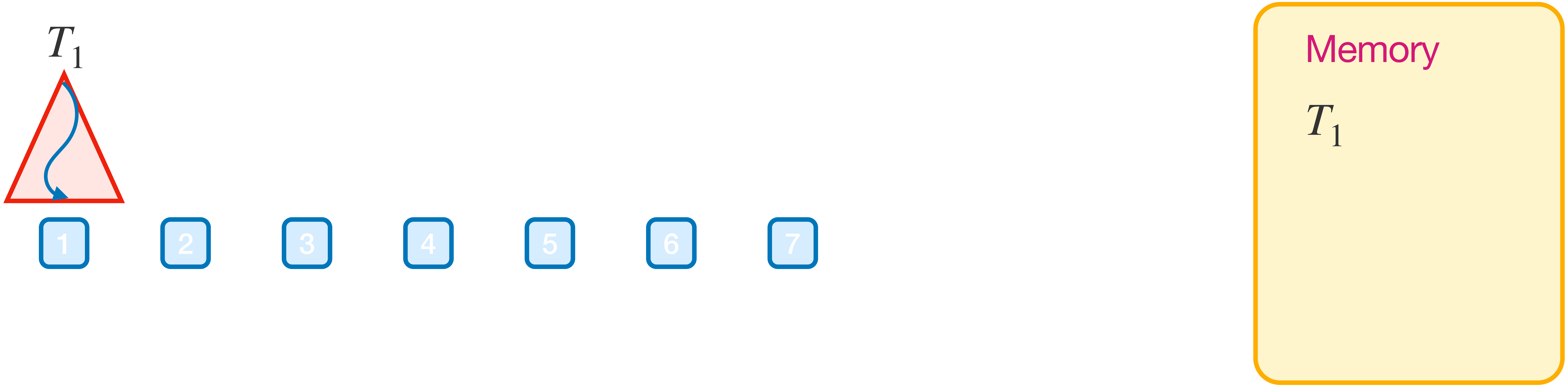
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$ Extract a Prover Strategy for $Search_F$



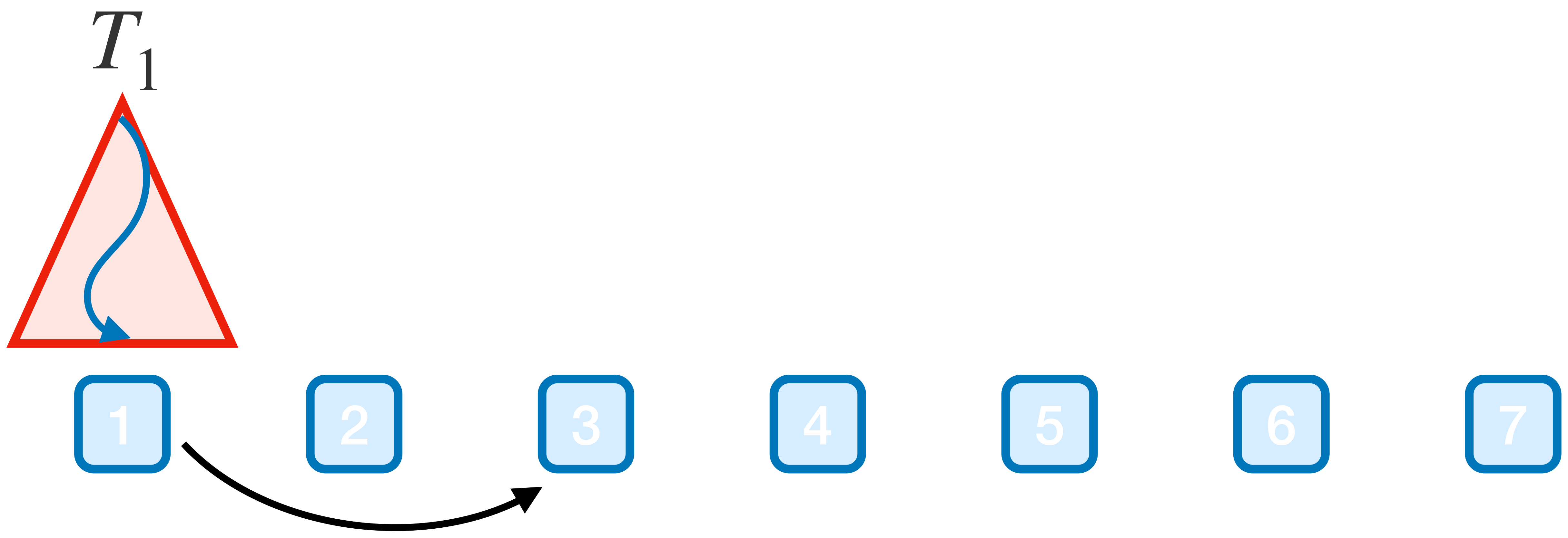
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$ Extract a Prover Strategy for $Search_F$



Memory

T_1

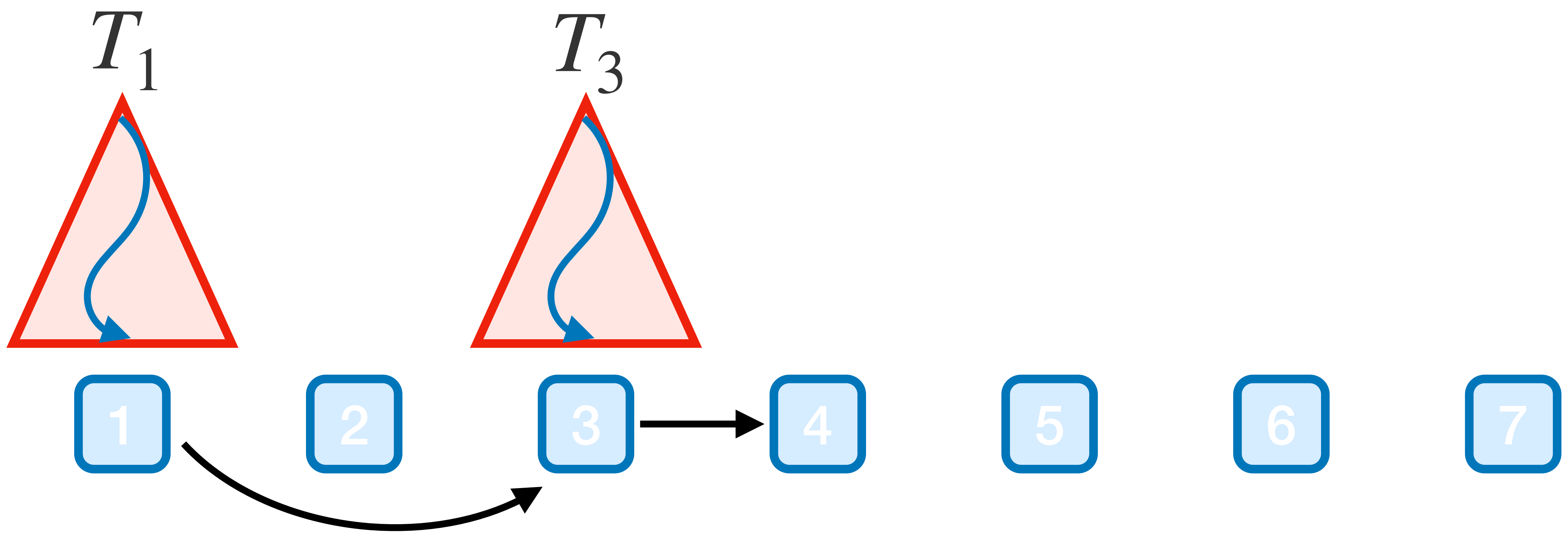
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$ Extract a Prover Strategy for $Search_F$



Memory
T_1
T_3

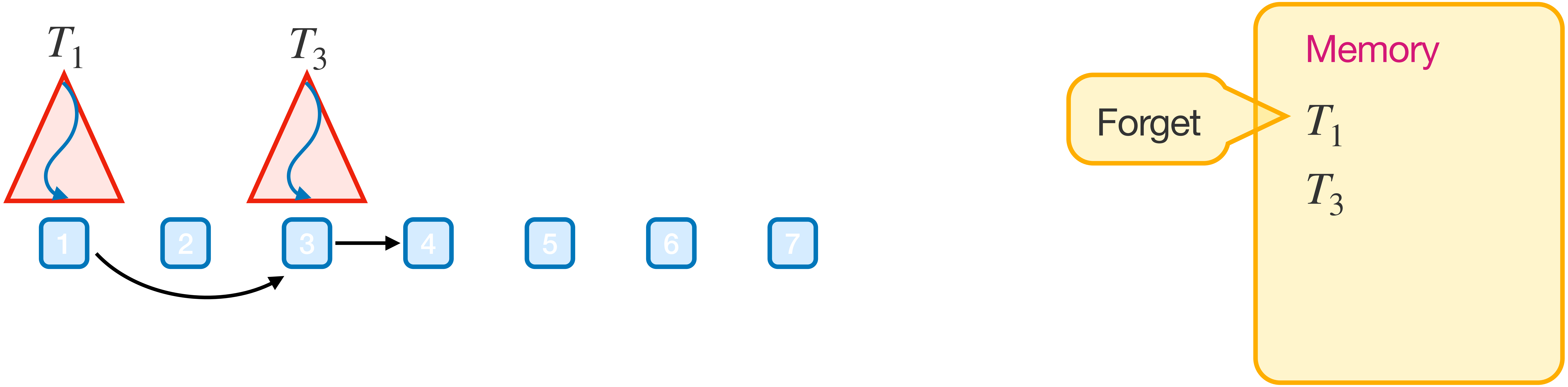
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$ Extract a Prover Strategy for $Search_F$



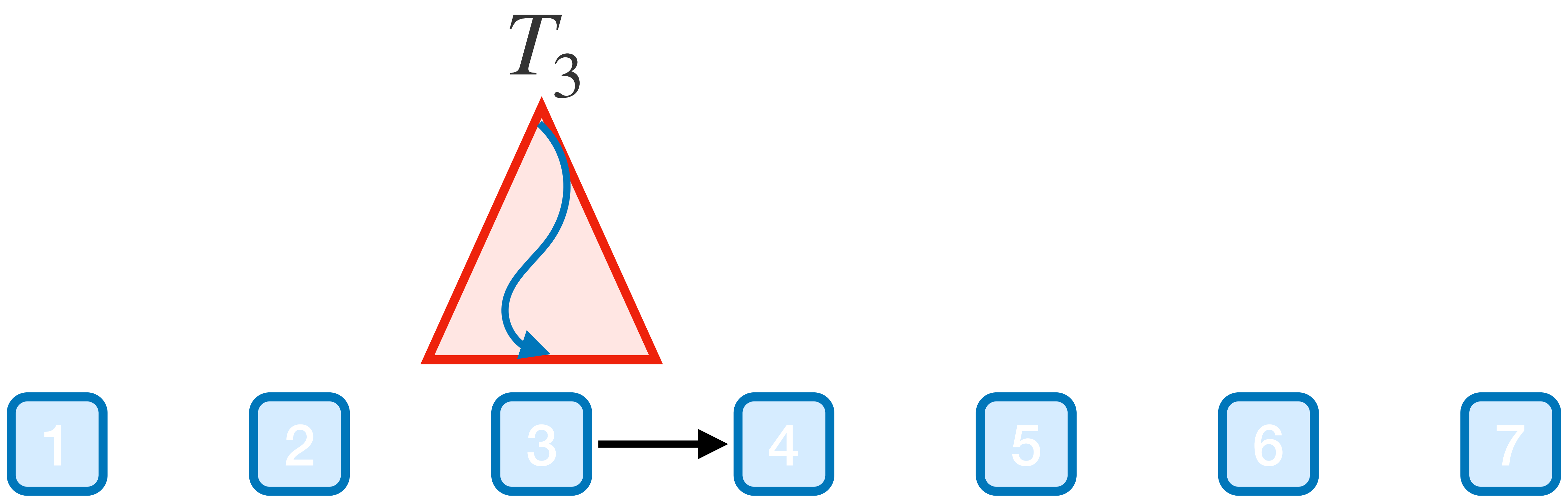
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$ Extract a Prover Strategy for $Search_F$



Memory

T_3

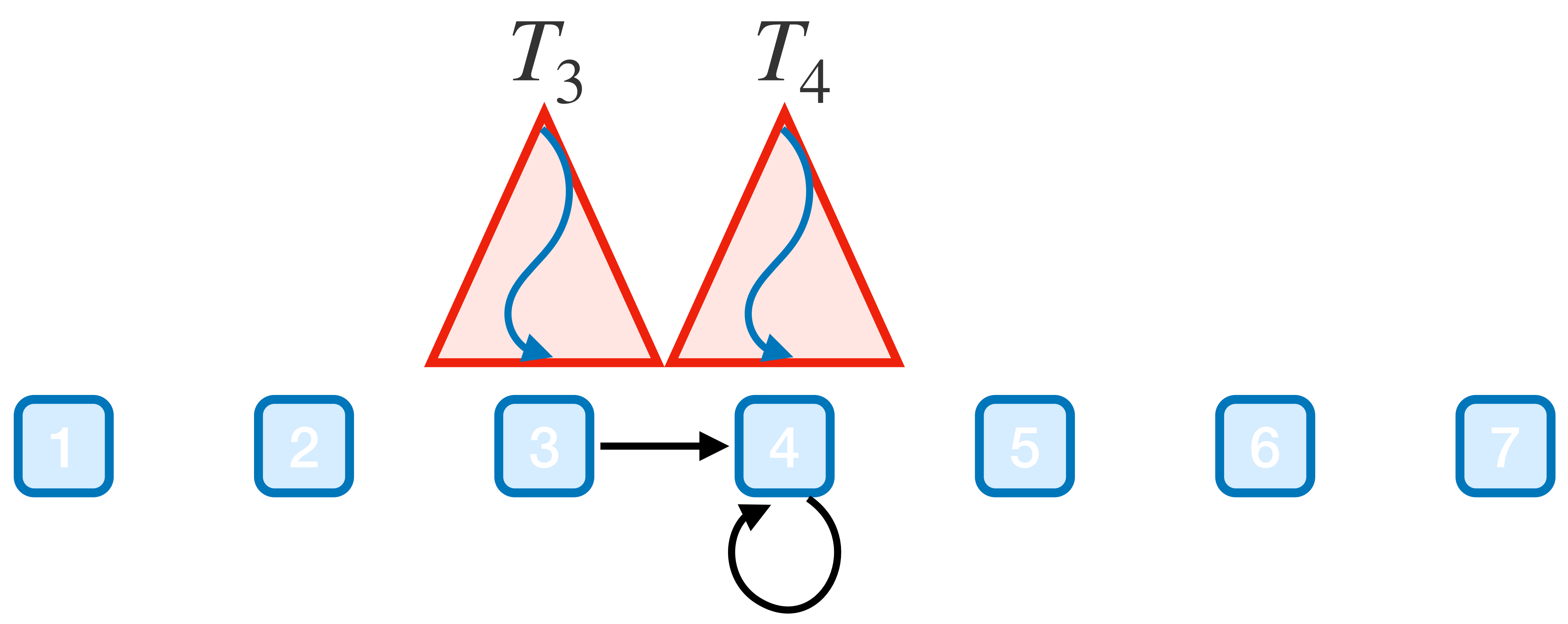
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$ Extract a Prover Strategy for $Search_F$



Memory
T_3
T_4

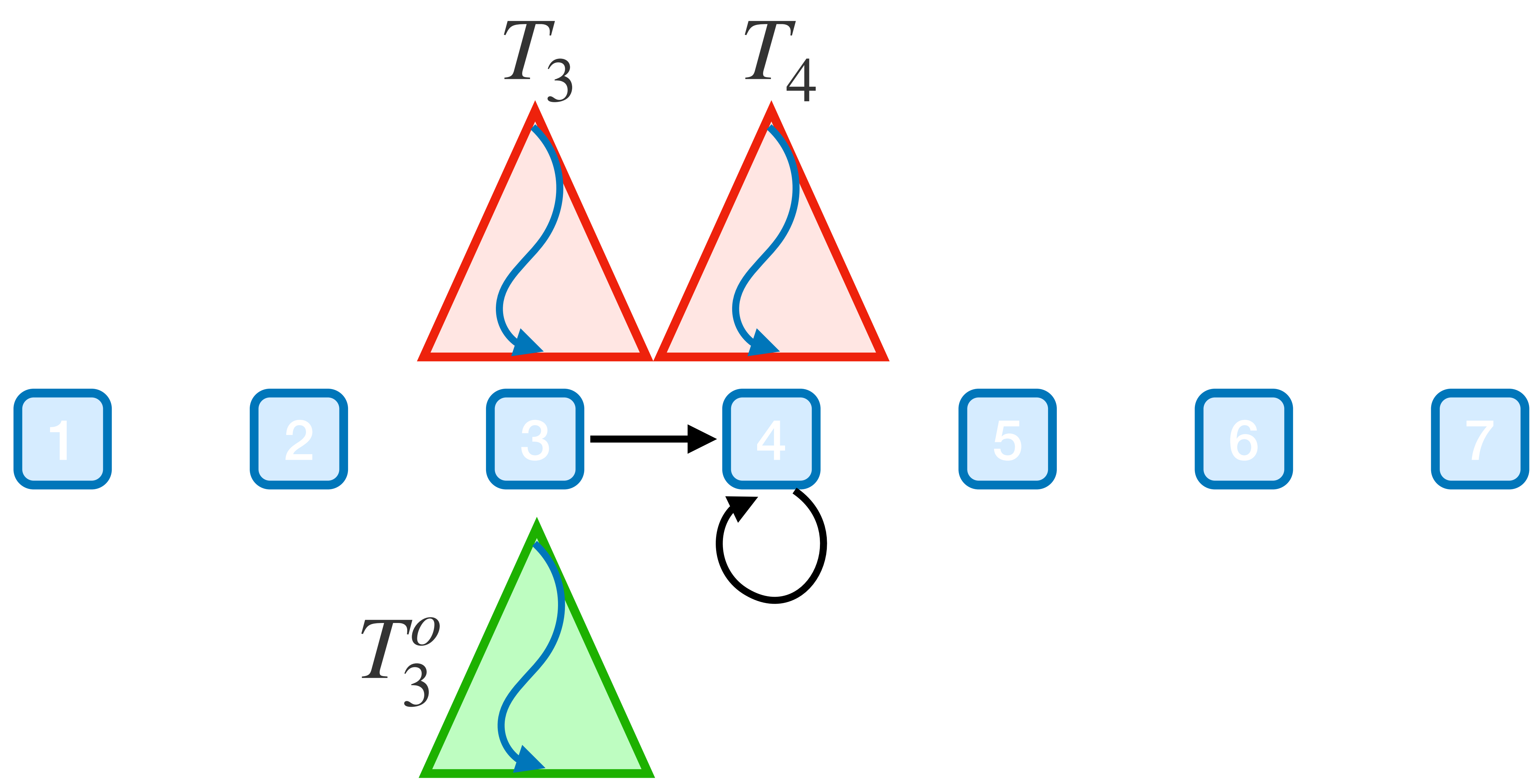
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$ Extract a Prover Strategy for $Search_F$



Memory
T_3
T_4
T_3^o

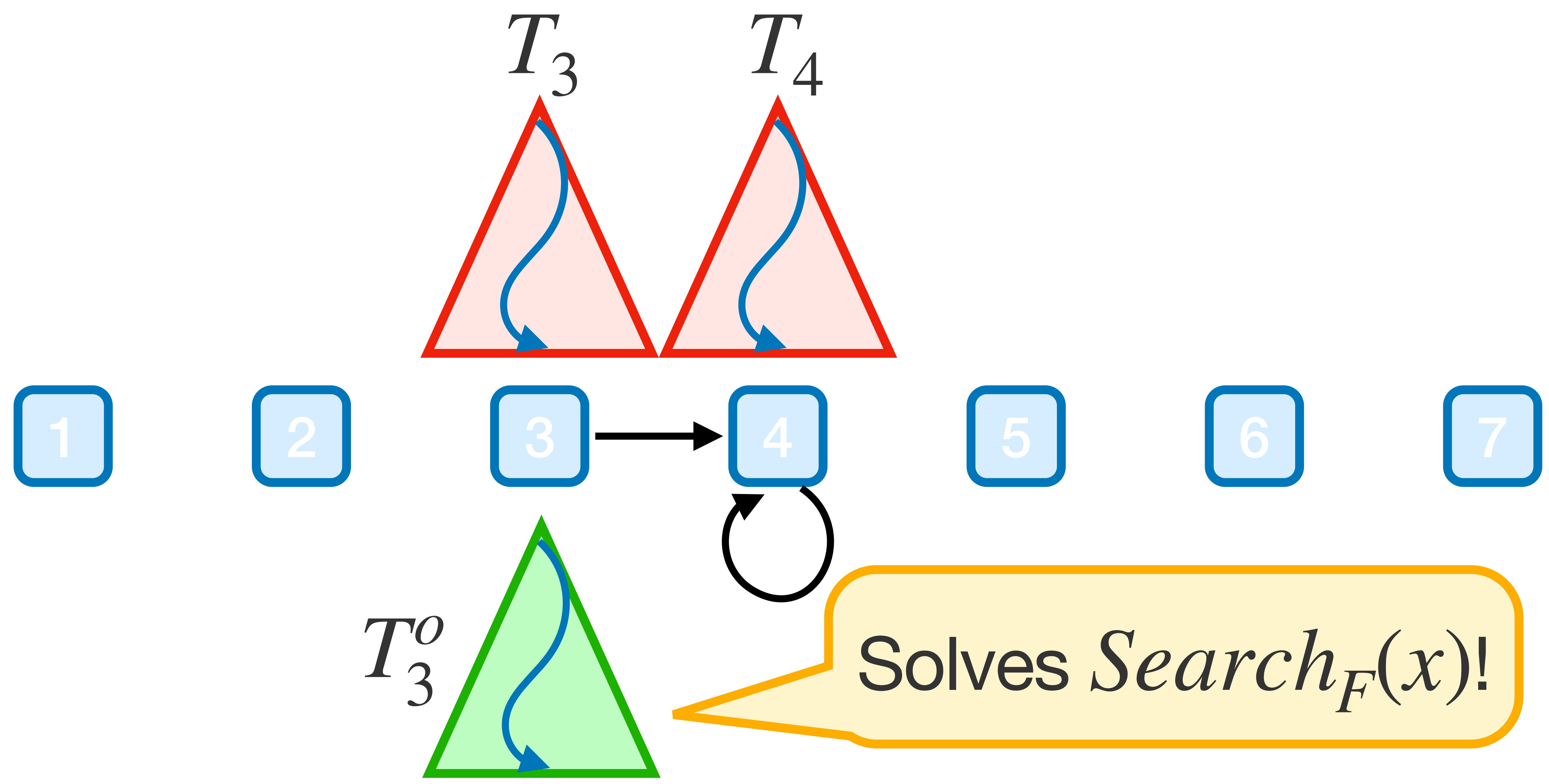
Resolution is PLS

Resolution Complexity: of proof Π is $\log size(\Pi) + width(\Pi)$

PLS is Resolution:
$PLS^{dt} = \{F : F$ has a polylog(n)-complexity Res proof$\}$

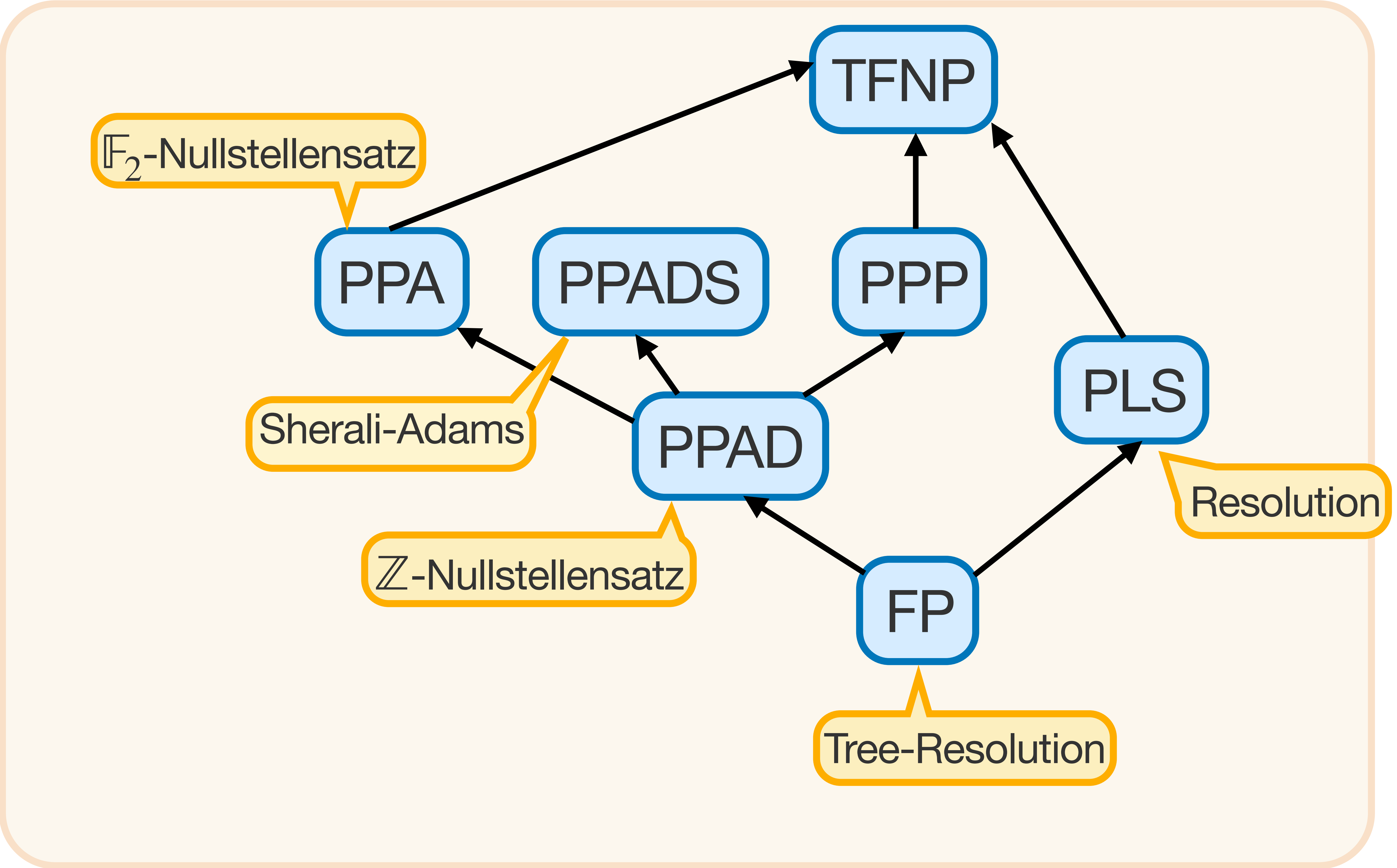
SinkOfDag
Vertices: $1, \ldots, n$
Pointers: $s_i \geq i$ with $s_1 \neq 1$
Solutions: i s.t. $s_i \neq i$ & $s_{s_i} = s_i$

$\Longleftarrow$ Extract a Prover Strategy for $Search_F$



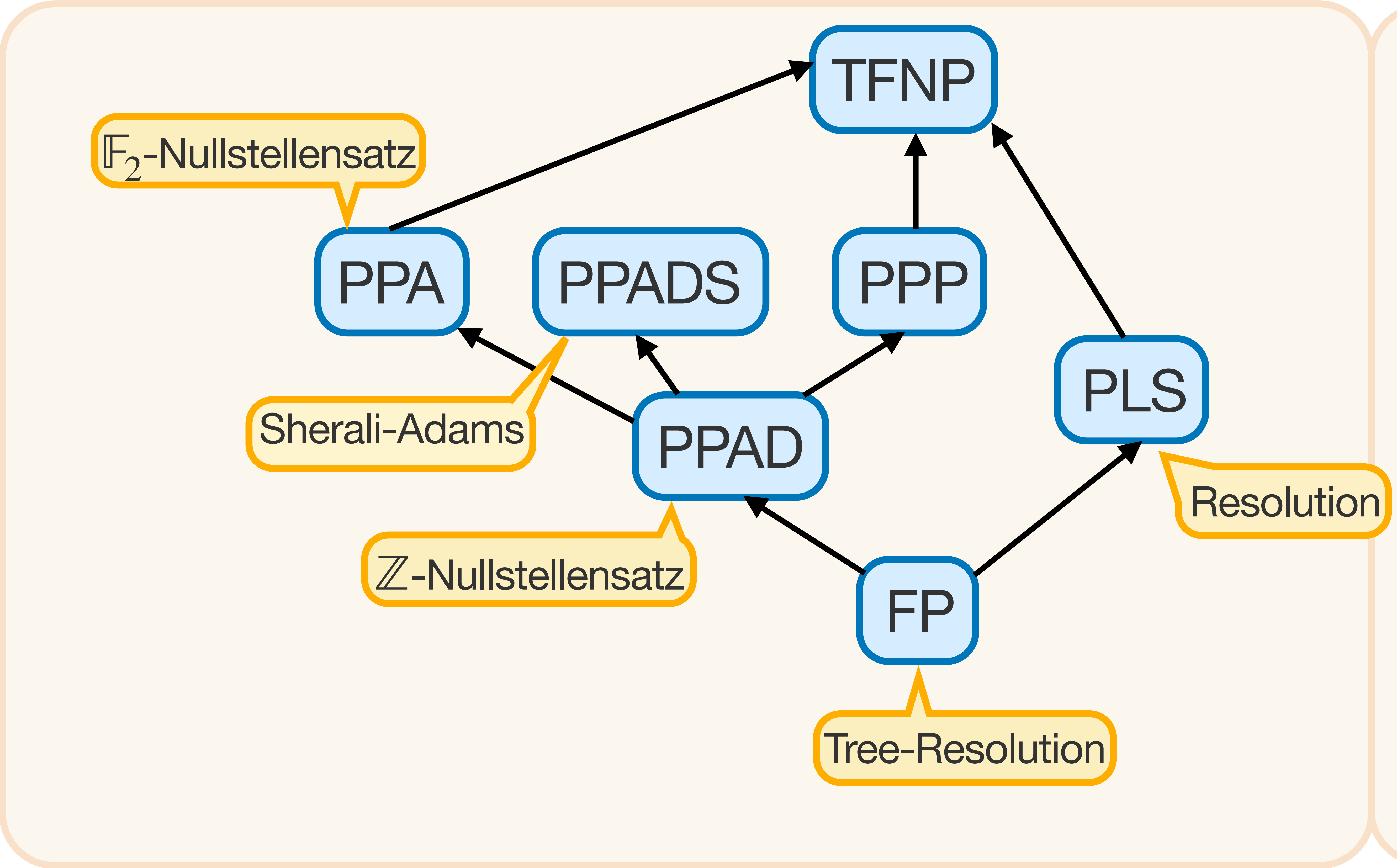
Memory
T_3
T_4
T_3^o

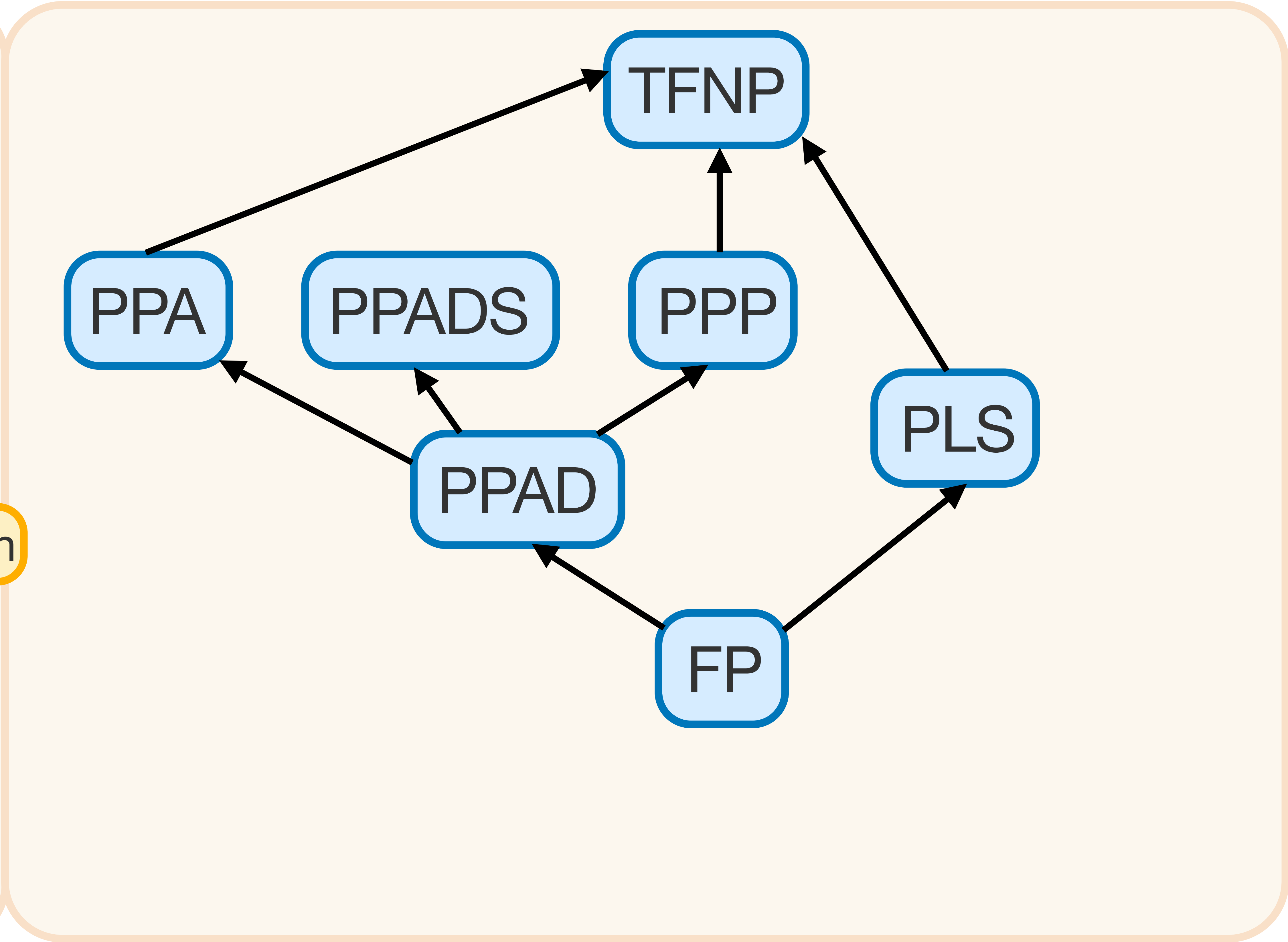
TFNP



Model of Computation: Decision Trees

TFNP

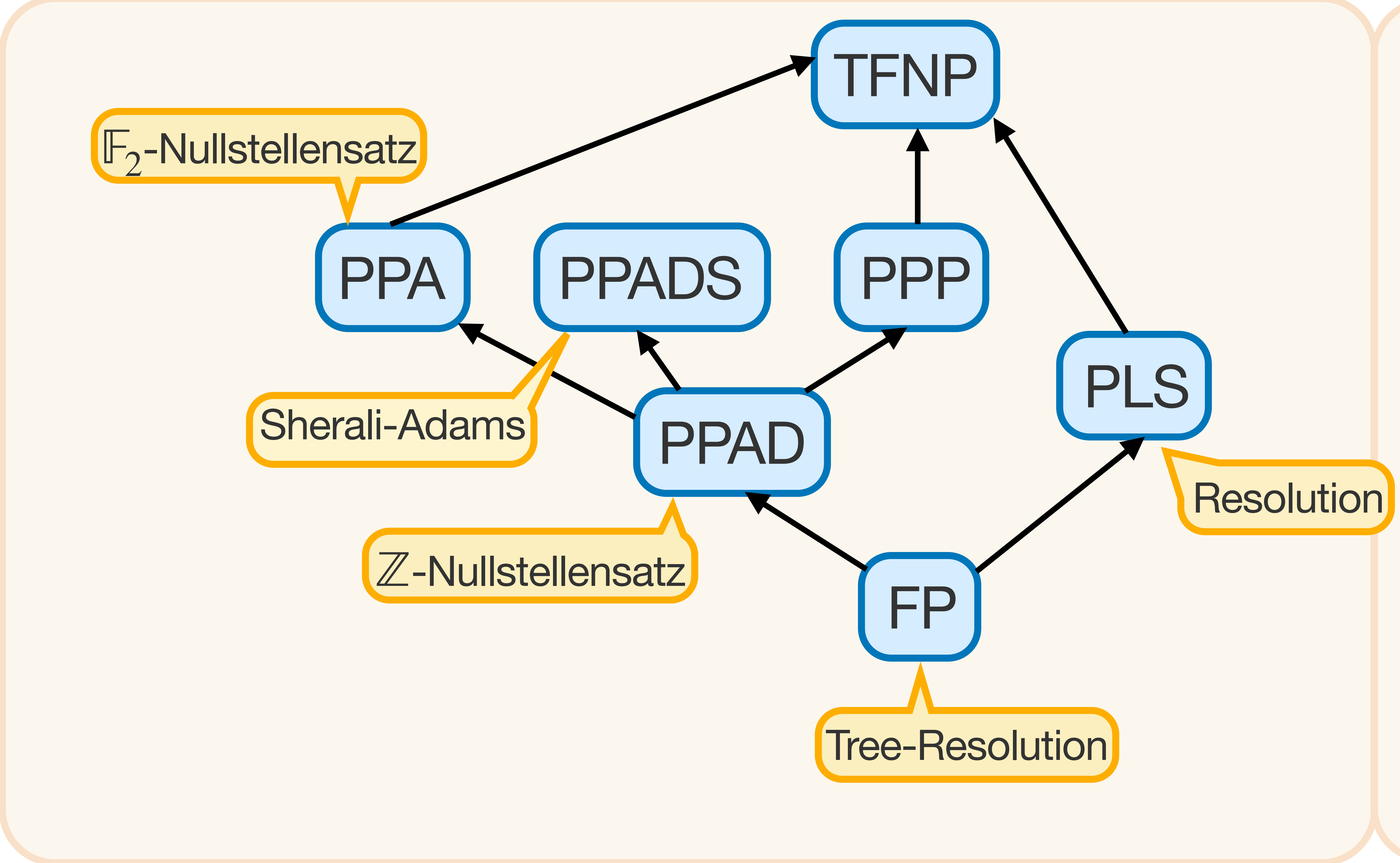


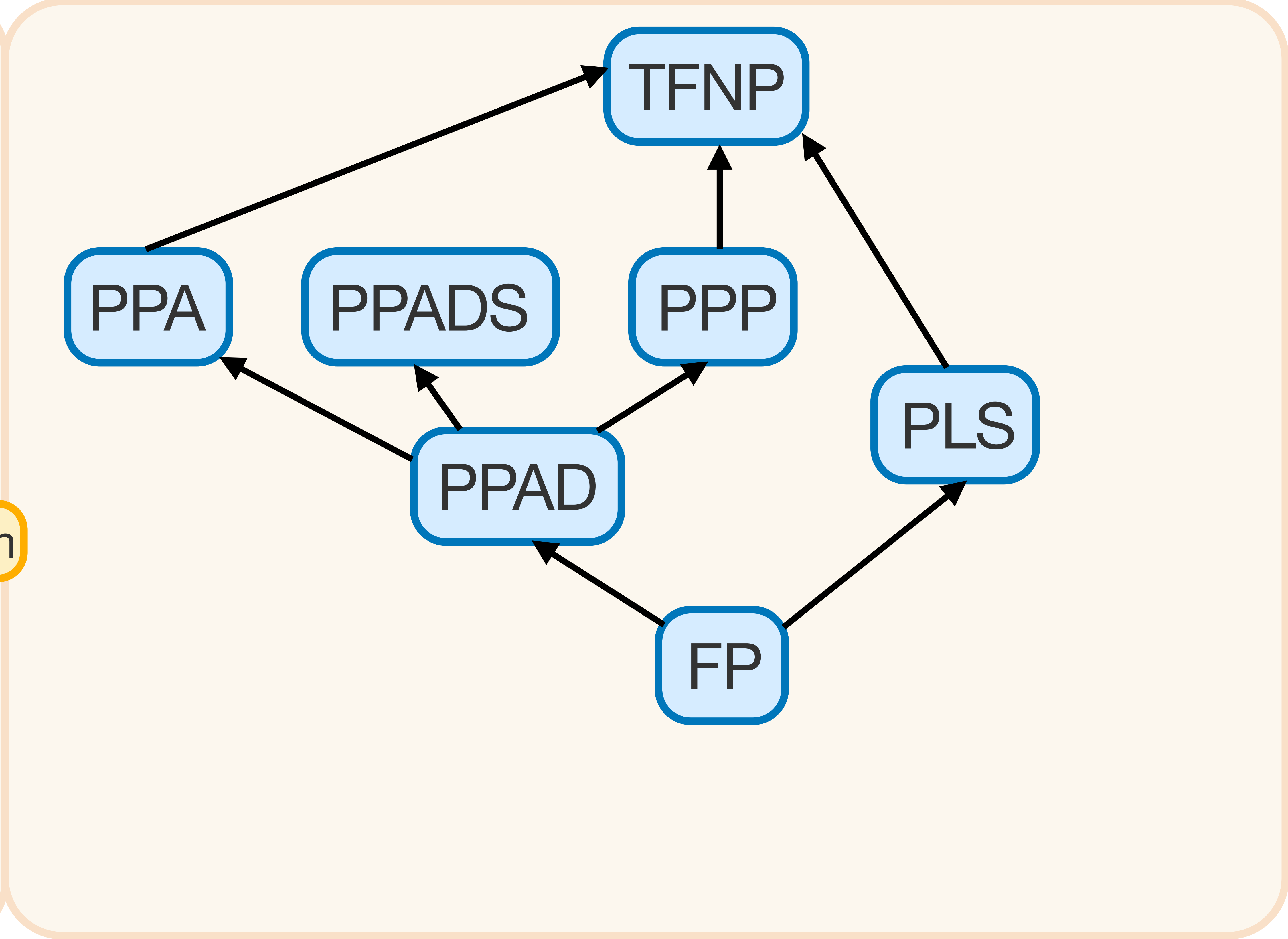


Model of Computation: Decision Trees Communication Protocols

TFNP

[GKRS18] Certain circuit models are **equivalent** to communication TFNP classes!

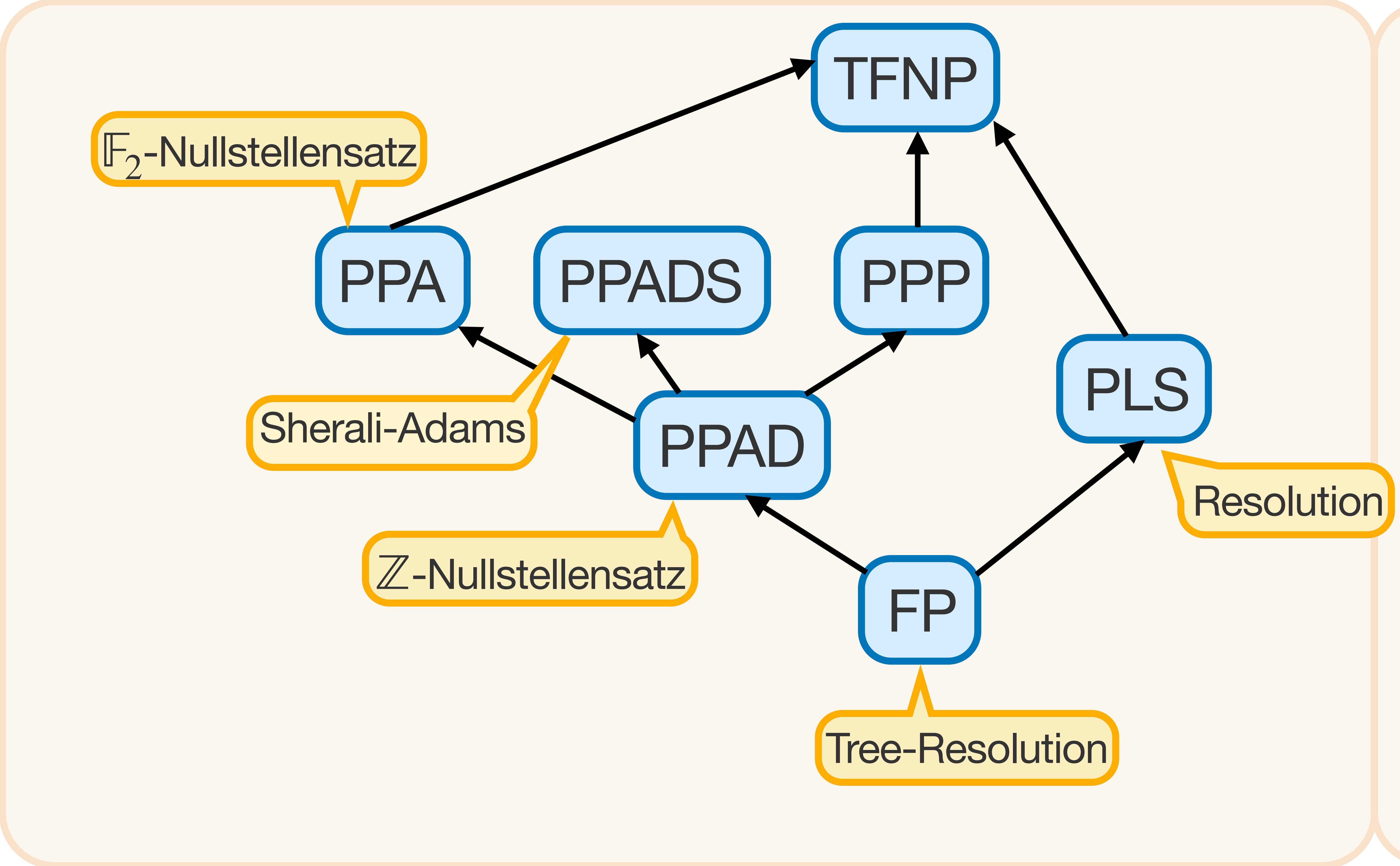


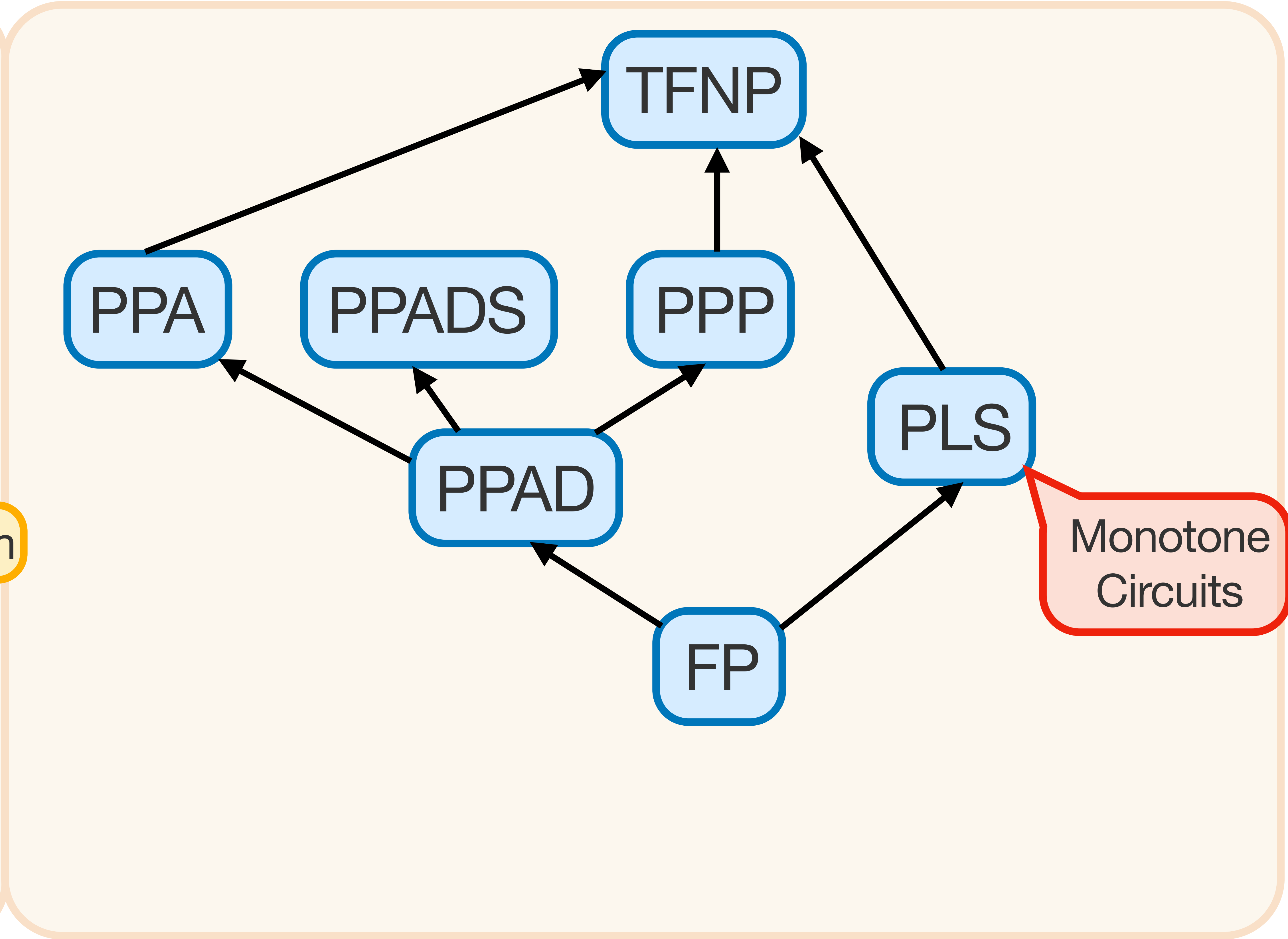


Model of Computation: Decision Trees Communication Protocols

TFNP

[GKRS18] Certain circuit models are **equivalent** to communication TFNP classes!

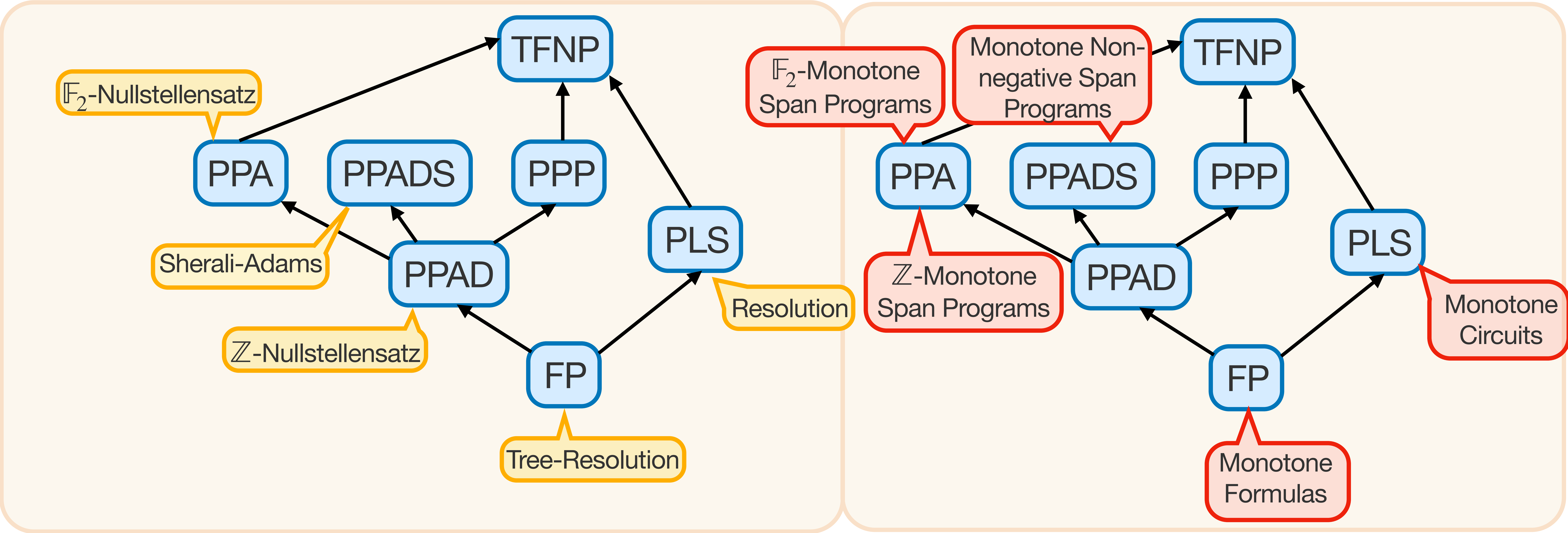




Model of Computation: Decision Trees Communication Protocols

TFNP

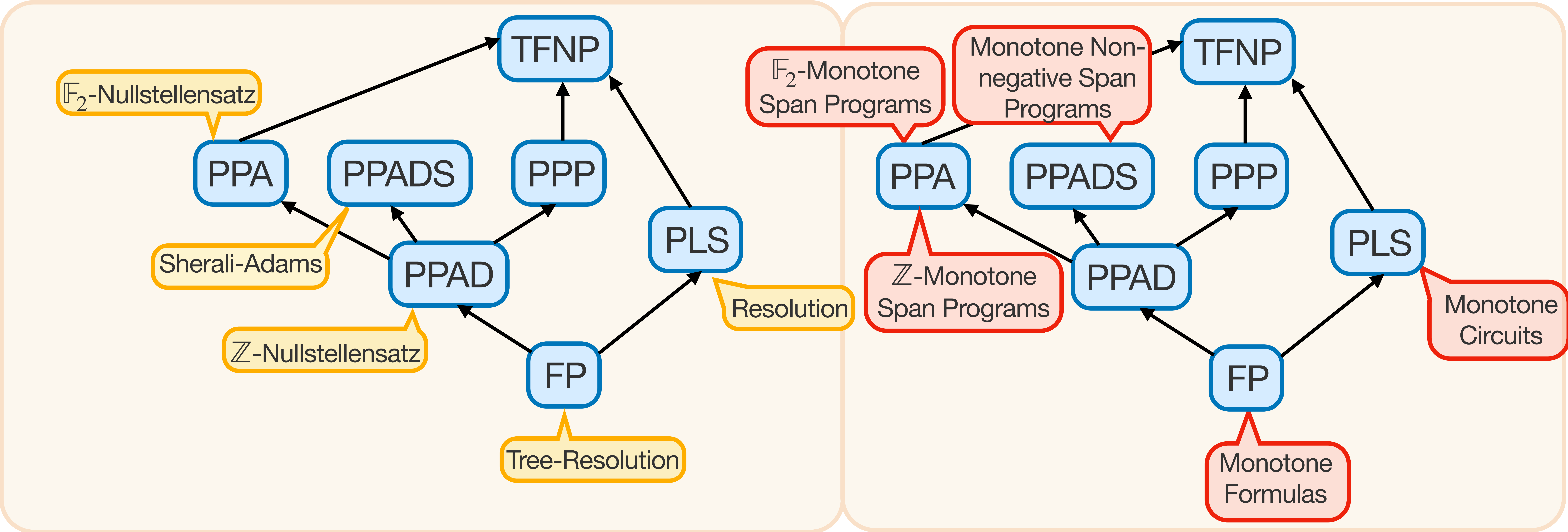
[GKRS18] Certain circuit models are **equivalent** to communication TFNP classes!



Model of Computation: Decision Trees Communication Protocols

TFNP

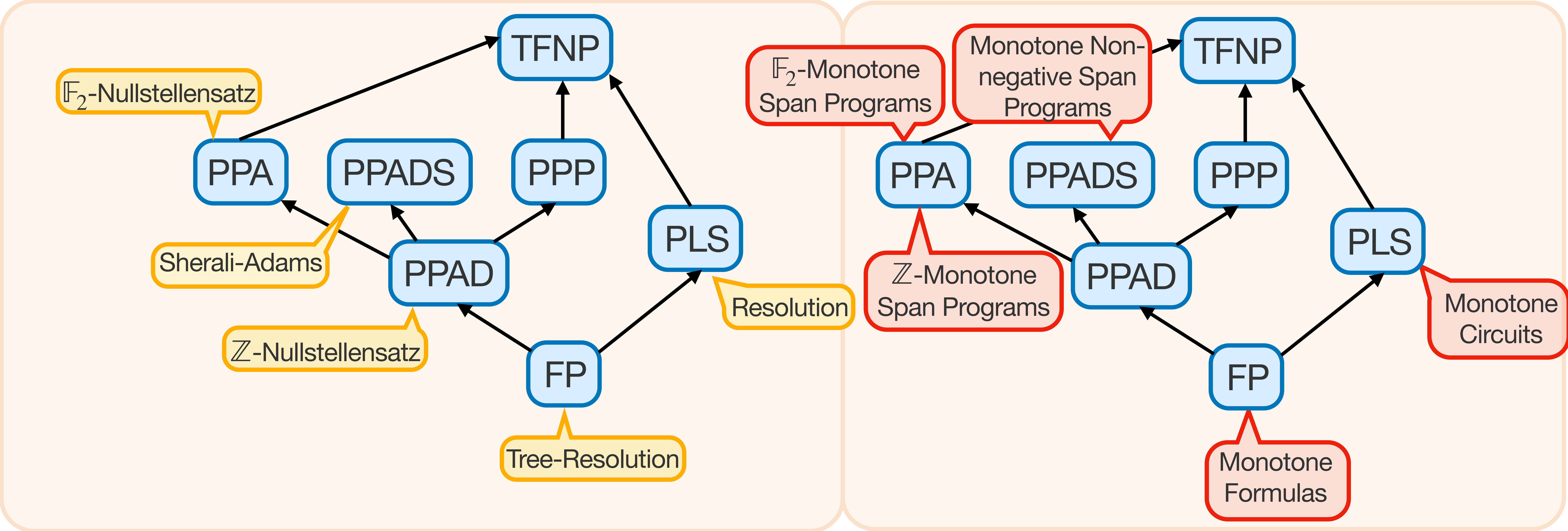
Observation 1: When both the DT and CC versions of a TFNP class admit a characterization then we immediately get an interpolation theorem



Model of Computation: Decision Trees | Communication Protocols

TFNP

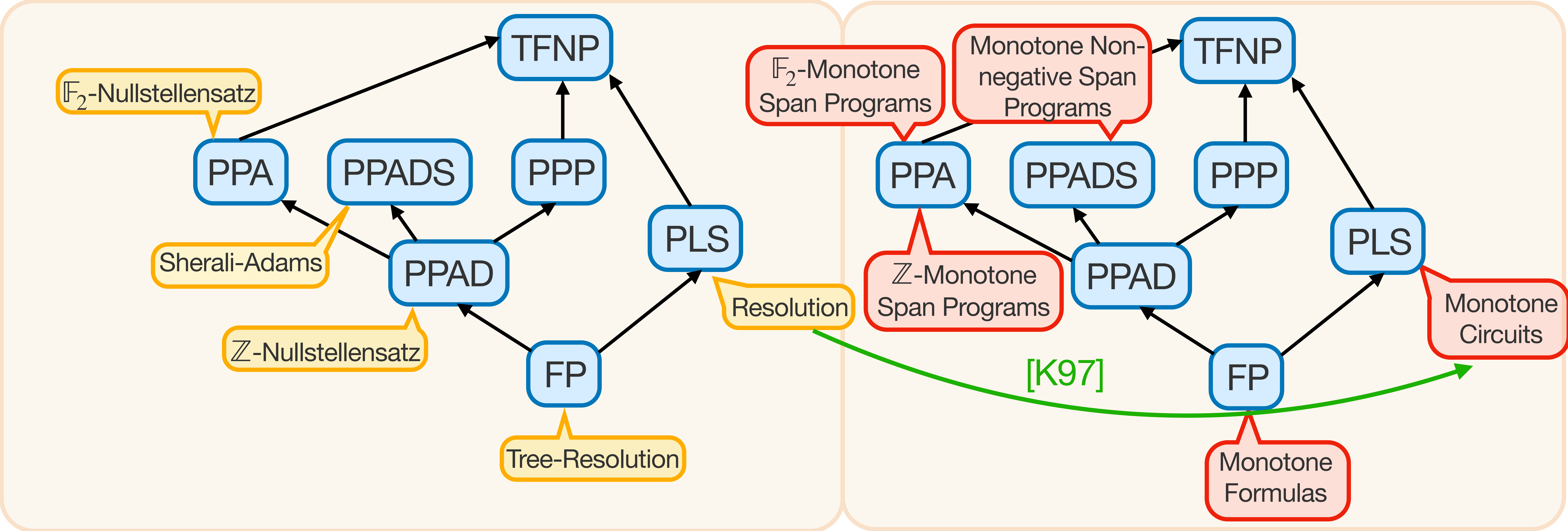
Observation 1: When both the DT and CC versions of a TFNP class admit a characterization then we immediately get an interpolation theorem — CC protocols can simulate DTs



Model of Computation: Decision Trees Communication Protocols

TFNP

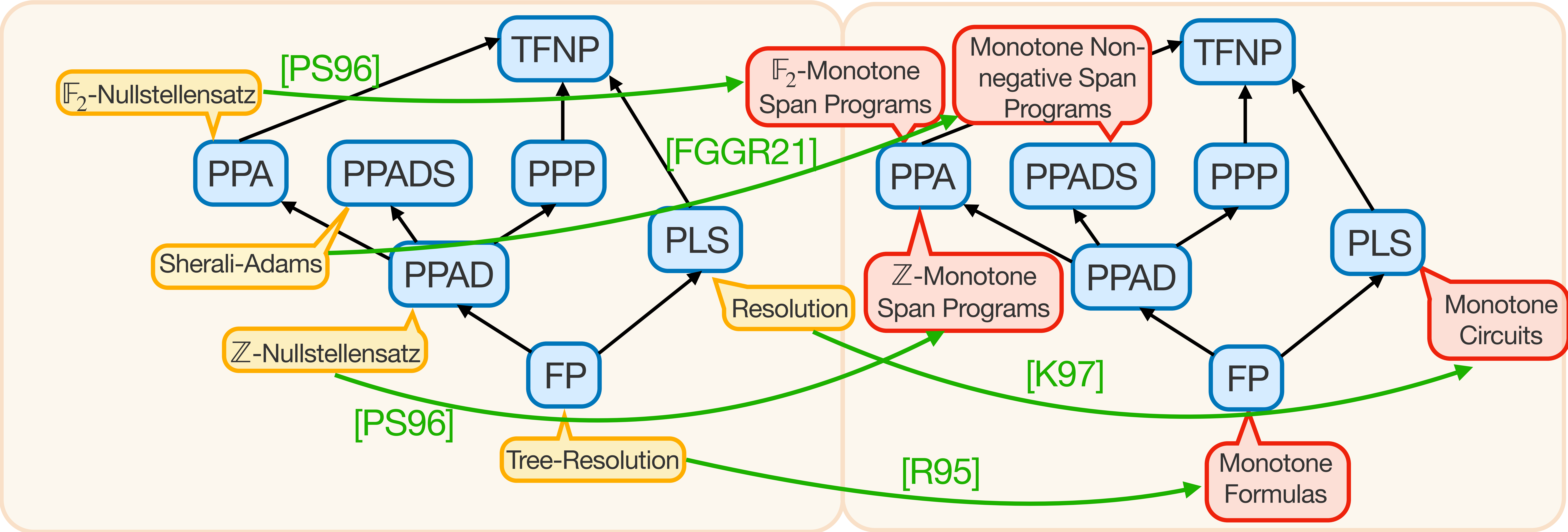
Observation 1: When both the DT and CC versions of a TFNP class admit a characterization then we immediately get an interpolation theorem — CC protocols can simulate DTs



Model of Computation: Decision Trees Communication Protocols

TFNP

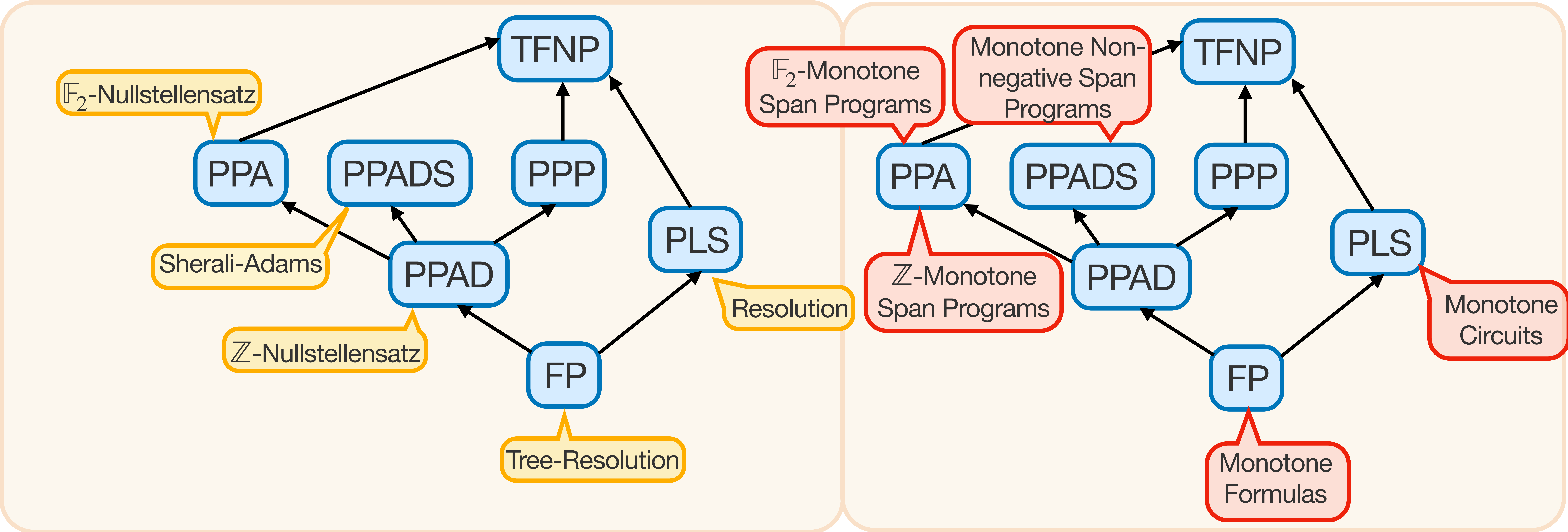
Observation 1: When both the DT and CC versions of a TFNP class admit a characterization then we immediately get an interpolation theorem — CC protocols can simulate DTs



Model of Computation: Decision Trees Communication Protocols

TFNP

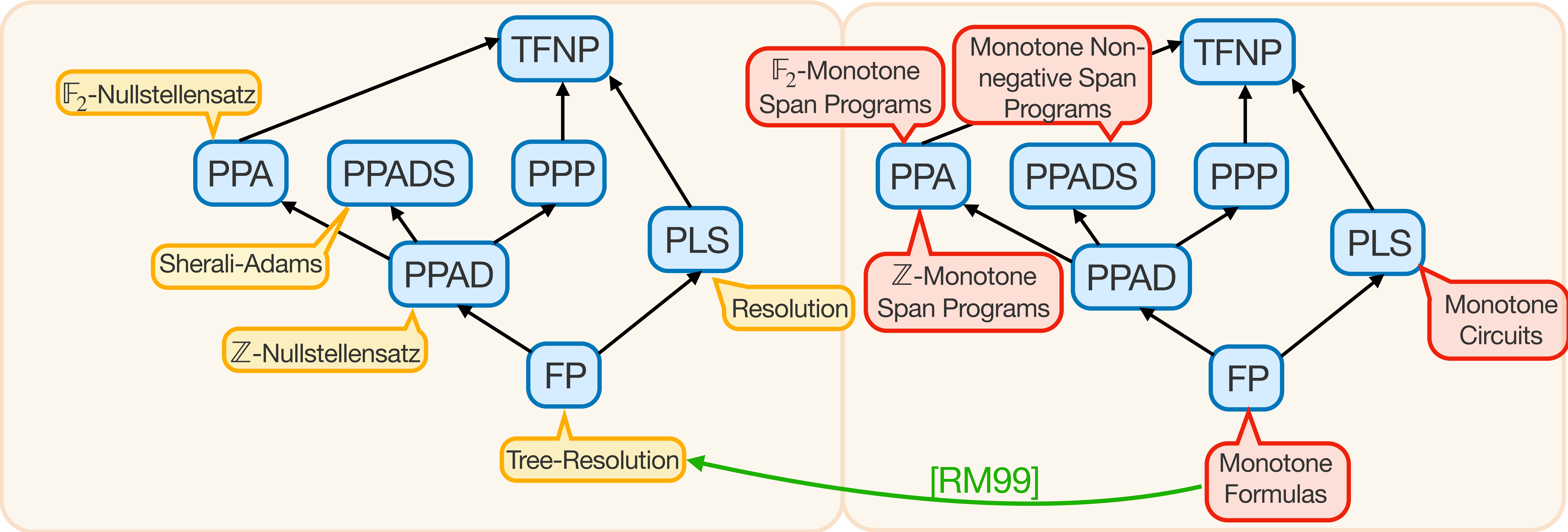
Observation 2: Query-to-communication lifting theorems can be viewed as relating the CC version of a TFNP class to its DT version.



Model of Computation: Decision Trees Communication Protocols

TFNP

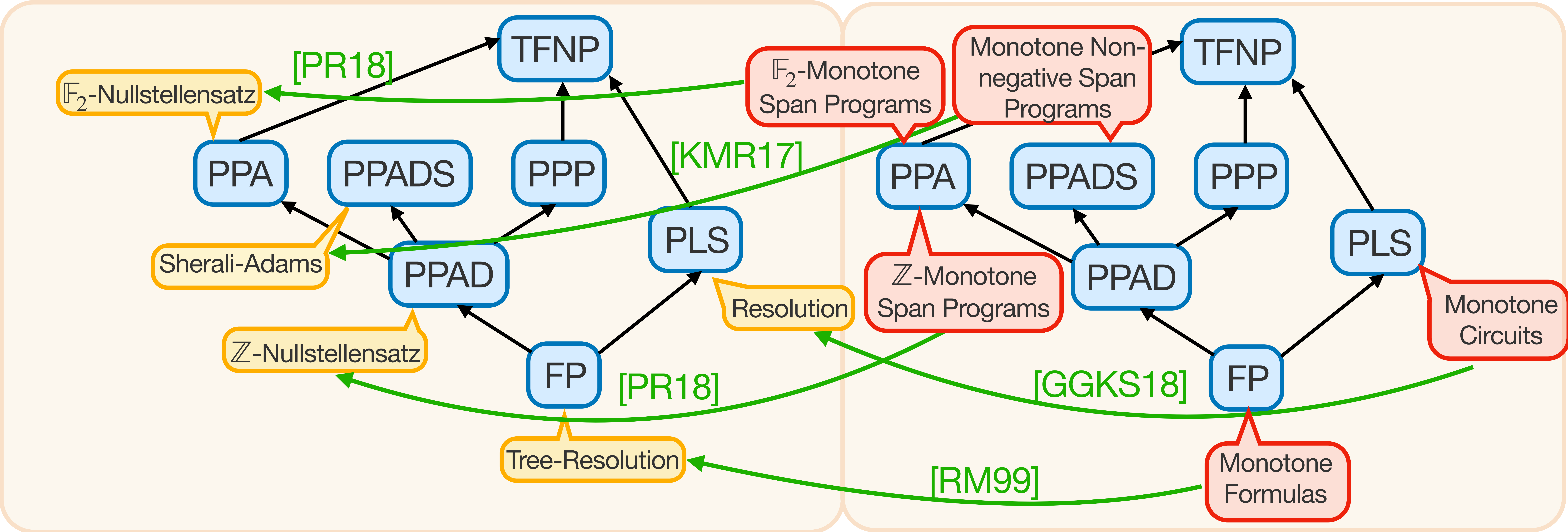
Observation 2: Query-to-communication lifting theorems can be viewed as relating the CC version of a TFNP class to its DT version.



Model of Computation: Decision Trees Communication Protocols

TFNP

Observation 2: Query-to-communication lifting theorems can be viewed as relating the CC version of a TFNP class to its DT version.



Model of Computation: Decision Trees Communication Protocols

TFNP: Interpolation & Lifting

Observation 1: When the DT and CC versions of a TFNP class both admit a characterization then we immediately get an interpolation theorem.

Observation 2: Query-to-communication lifting theorems can be viewed as relating the CC version of a TFNP class to its DT version.

TFNP: Interpolation & Lifting

Observation 1: When the DT and CC versions of a TFNP class both admit a characterization then we immediately get an interpolation theorem.

Observation 2: Query-to-communication lifting theorems can be viewed as relating the CC version of a TFNP class to its DT version.

Upshot: Understand when interpolation or query-to-communication lifting theorems occur by understanding when proof systems and monotone circuit models admit TFNP characterizations!

TFNP: Interpolation & Lifting

Observation 1: When the DT and CC versions of a TFNP class both admit a characterization then we immediately get an interpolation theorem.

Observation 2: Query-to-communication lifting theorems can be viewed as relating the CC version of a TFNP class to its DT version.

Upshot: Understand when interpolation or query-to-communication lifting theorems occur by understanding when proof systems and monotone circuit models admit TFNP characterizations!

$\mathcal{Q}$. Under what conditions does a TFNP class admit a proof system / circuit characterization?

TFNP: Interpolation & Lifting

Observation 1: When the DT and CC versions of a TFNP class both admit a characterization then we immediately get an interpolation theorem.

Observation 2: Query-to-communication lifting theorems can be viewed as relating the CC version of a TFNP class to its DT version.

Upshot: Understand when interpolation or query-to-communication lifting theorems occur by understanding when proof systems and monotone circuit models admit TFNP characterizations!

$\mathcal{Q}$. Under what conditions does a TFNP class admit a proof system / circuit characterization?

$\mathcal{Q}$. Under what conditions does a proof system / circuit admit a TFNP characterization?

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

$\rightarrow$ Proofs are reductions to a complete problem for C!

Proof Complexity Characterizations

$Q.$ Under what conditions does a TFNP class admit a proof system characterization?

$A.$ For every TFNP class C there is a proof system which characterizes it!

→ Proofs are reductions to a complete problem for C!

Canonical proof system for C

Fix H such that $Search_H$ is equivalent to the complete problem for C

Proof Complexity Characterizations

$Q.$ Under what conditions does a TFNP class admit a proof system characterization?

$A.$ For every TFNP class C there is a proof system which characterizes it!

→ Proofs are reductions to a complete problem for C!

Canonical proof system for C

Fix H such that $Search_H$ is equivalent to the complete problem for C

Proof of F: a tuple $(n', \{T_i\}, \{T_j^o\})$ which describes a reduction from $Search_F$ to $Search_H$ on n' variables.

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

→ Proofs are reductions to a complete problem for C!

Canonical proof system for C

Fix H such that $Search_H$ is equivalent to the complete problem for C

Proof of F: a tuple $(n', \{T_i\}, \{T_j^o\})$ which describes a reduction from $Search_F$ to $Search_H$ on n' variables.

Cook-Reckhow proof system — proofs are verifiable!

→ Just check that $(n', \{T_i\}, \{T_j^o\})$ describes a valid reduction!

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

$\rightarrow$ Proofs are reductions to a complete problem for C!

Q. Under what conditions does a proof system admit a TFNP characterization?

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

→ Proofs are reductions to a complete problem for C!

Q. Under what conditions does a proof system admit a TFNP characterization?

Proof Complexity Characterizations

$Q.$ Under what conditions does a TFNP class admit a proof system characterization?

$A.$ For every TFNP class C there is a proof system which characterizes it!

→ Proofs are reductions to a complete problem for C!

$Q.$ Under what conditions does a proof system admit a TFNP characterization?

$A.$ Iff the proof system P:

- has short proofs of its own soundness!

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

$\rightarrow$ Proofs are reductions to a complete problem for C!

Q. Under what conditions does a proof system admit a TFNP characterization?

A. Iff the proof system P:

- has short proofs of its own soundness!

Efficiently verifiable version of a reflection principle about itself

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

→ Proofs are reductions to a complete problem for C!

Q. Under what conditions does a proof system admit a TFNP characterization?

A. Iff the proof system P:

- has short proofs of its own soundness!

Efficiently verifiable version of a reflection principle about itself

"If F has a P-proof then F is a tautology"

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

→ Proofs are reductions to a complete problem for C!

Q. Under what conditions does a proof system admit a TFNP characterization?

A. Iff the proof system P:

- has short proofs of its own soundness!

Efficiently verifiable version of a reflection principle about itself

polylog-width

"If F has a P-proof then F is a tautology"

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

→ Proofs are reductions to a complete problem for C!

Q. Under what conditions does a proof system admit a TFNP characterization?

A. Iff the proof system P:

- has short proofs of its own soundness!
- Closed under dt-reductions

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

→ Proofs are reductions to a complete problem for C!

Q. Under what conditions does a proof system admit a TFNP characterization?

A. Iff the proof system P:

- has short proofs of its own soundness!
- Closed under dt-reductions

If P has a small proof of F and $T_1, \ldots, T_n$ are short decision trees
$\implies P$ has a small proof of $F(T_1, \ldots, T_n)$

Proof Complexity Characterizations

Q. Under what conditions does a TFNP class admit a proof system characterization?

A. For every TFNP class C there is a proof system which characterizes it!

→ Proofs are reductions to a complete problem for C!

Q. Under what conditions does a proof system admit a TFNP characterization?

A. Iff the proof system P:

- has short proofs of its own soundness!
- Closed under dt-reductions

If P has a small proof of F and $T_1, \ldots, T_n$ are short decision trees
$\Longrightarrow P$ has a small proof of $F(T_1, \ldots, T_n)$

Standard proof systems satisfy this — e.g., Resolution, Sherali-Adams, Nullstellensatz…

Short Proofs of Soundness

Reflection principle for proof system P

$$Ref_{P,n,m,c} := Proof_P(F, \Pi) \wedge SAT(F, \alpha)$$

Short Proofs of Soundness

Reflection principle for proof system P

$$Ref_{P,n,m,c} := Proof_P(F, \Pi) \wedge SAT(F, \alpha)$$

Short Proofs of Soundness

Reflection principle for proof system P

Π is a complexity-c P-proof that F is unsatisfiable

$$Ref_{P,n,m,c} := Proof_P(F, \Pi) \wedge SAT(F, \alpha)$$

Short Proofs of Soundness

Reflection principle for proof system P

$$Ref_{P,n,m,c} := Proof_P(F, \Pi) \wedge SAT(F, \alpha)$$

Π is a complexity-c P-proof that F is unsatisfiable

α is a satisfying assignment for F

Short Proofs of Soundness

Reflection principle for proof system P

Π is a complexity-c P-proof that F is unsatisfiable

$$Ref_{P,n,m,c} := Proof_P(F, \Pi) \wedge SAT(F, \alpha)$$

α is a satisfying assignment for F

Fix a standard encoding of SAT

Short Proofs of Soundness

Reflection principle for proof system P

Π is a complexity-c P-proof that F is unsatisfiable

$$Ref_{P,n,m,c} := Proof_P(F, \Pi) \wedge SAT(F, \alpha)$$

α is a satisfying assignment for F

Fix a standard encoding of SAT

Many ways to encode P-proofs in an efficiently verifiable manner ($O(c)$ width, $2^{O(c)}$ size)

Short Proofs of Soundness

Reflection principle for proof system P

Π is a complexity-c P-proof that F is unsatisfiable

$$Ref_{P,n,m,c} := Proof_P(F, \Pi) \land SAT(F, \alpha)$$

α is a satisfying assignment for F

Fix a standard encoding of SAT

Many ways to encode P-proofs in an efficiently verifiable manner ($O(c)$ width, $2^{O(c)}$ size)

→ Each generates a TFNP class as everything reducible to $Search_{Ref_P}$

Efficiently Verifiable Reflection Principles

Theorem: If P is closed under dt-reductions and has $polylog(n)$-complexity proofs of Ref_P then P is characterized by the TFNP class for $Search_{Ref_P}$

Efficiently Verifiable Reflection Principles

Theorem: If P is closed under dt-reductions and has $polylog(n)$-complexity proofs of Ref_P then P is characterized by the TFNP class for $Search_{Ref_P}$

$Search_{Ref_P} \in TFNP^{dt}$ as Ref_P is efficiently verifiable.

Efficiently Verifiable Reflection Principles

Theorem: If P is closed under dt-reductions and has $polylog(n)$-complexity proofs of Ref_P then P is characterized by the TFNP class for $Search_{Ref_P}$

$Search_{Ref_P} \in TFNP^{dt}$ as Ref_P is efficiently verifiable.

$Search_F$ reduces to $Search_{Ref_P} \implies$ efficient P-proof of F:

Efficient P-proof of $F \implies Search_F$ reduces to $Search_{Ref_F}$

Efficiently Verifiable Reflection Principles

Theorem: If P is closed under dt-reductions and has $polylog(n)$-complexity proofs of Ref_P then P is characterized by the TFNP class for $Search_{Ref_P}$

$Search_{Ref_P} \in TFNP^{dt}$ as Ref_P is efficiently verifiable.

$Search_F$ reduces to $Search_{Ref_P} \Longrightarrow$ efficient P-proof of F:

As P is closed under dt-reductions and has a short proof of Ref_P then it has a short proof of F

Efficient P-proof of $F \Longrightarrow Search_F$ reduces to $Search_{Ref_F}$

Efficiently Verifiable Reflection Principles

Theorem: If P is closed under dt-reductions and has $polylog(n)$-complexity proofs of Ref_P then P is characterized by the TFNP class for $Search_{Ref_P}$

$Search_{Ref_P} \in TFNP^{dt}$ as Ref_P is efficiently verifiable.

$Search_F$ reduces to $Search_{Ref_P} \Longrightarrow$ efficient P-proof of F:

As P is closed under dt-reductions and has a short proof of Ref_P then it has a short proof of F

Efficient P-proof of $F \Longrightarrow Search_F$ reduces to $Search_{Ref_F}$

Let Π be an efficient P-proof of F

Efficiently Verifiable Reflection Principles

Theorem: If P is closed under dt-reductions and has $polylog(n)$-complexity proofs of Ref_P then P is characterized by the TFNP class for $Search_{Ref_P}$

$Search_{Ref_P} \in TFNP^{dt}$ as Ref_P is efficiently verifiable.

$Search_F$ reduces to $Search_{Ref_P} \Longrightarrow$ efficient P-proof of F:

As P is closed under dt-reductions and has a short proof of Ref_P then it has a short proof of F

Efficient P-proof of $F \Longrightarrow Search_F$ reduces to $Search_{Ref_F}$

Let Π be an efficient P-proof of F

Reduction hardwires Π, F in $Ref_P(\Pi, F, \alpha)$ leaving only the assignment α free (using constant DTs)

Efficiently Verifiable Reflection Principles

Theorem: If P is closed under dt-reductions and has $polylog(n)$-complexity proofs of Ref_P then P is characterized by the TFNP class for $Search_{Ref_P}$

$Search_{Ref_P} \in TFNP^{dt}$ as Ref_P is efficiently verifiable.

$Search_F$ reduces to $Search_{Ref_P} \implies$ efficient P-proof of F:

As P is closed under dt-reductions and has a short proof of Ref_P then it has a short proof of F

Efficient P-proof of $F \implies Search_F$ reduces to $Search_{Ref_F}$

Let Π be an efficient P-proof of F

Reduction hardwires Π, F in $Ref_P(\Pi, F, \alpha)$ leaving only the assignment α free (using constant DTs)

Π is low complexity $\implies$ number of variables of Ref_P instance is not much more than that of F

Efficiently Verifiable Reflection Principles

Theorem: If P is closed under dt-reductions and has $polylog(n)$-complexity proofs of Ref_P then P is characterized by the TFNP class for $Search_{Ref_P}$

Efficiently Verifiable Reflection Principles

Theorem: If P is closed under dt-reductions and has $polylog(n)$-complexity proofs of Ref_P then P is characterized by the TFNP class for $Search_{Ref_P}$

Canonical proof system for a TFNP class can prove a reflection principle about itself

Efficiently Verifiable Reflection Principles

Theorem: If P is closed under dt-reductions and has $polylog(n)$-complexity proofs of Ref_P then P is characterized by the TFNP class for $Search_{Ref_P}$

Canonical proof system for a TFNP class can prove a reflection principle about itself

Corollary: A proof system admits a TFNP^{dt} characterization iff it is closed under decision tree reductions and has short proofs of a reflection principle about itself.

Circuit Complexity

Q. Under what conditions does a TFNP class admit a circuit characterization?

Circuit Complexity

Q. Under what conditions does a TFNP class admit a circuit characterization?

A. For every TFNP class there is a model of monotone circuit which characterizes it!

Circuit Complexity

Q. Under what conditions does a TFNP class admit a circuit characterization?

A. For every TFNP class there is a model of monotone circuit which characterizes it!

Q. Under what conditions does a monotone circuit model admit a TFNP characterization?

Circuit Complexity

Q. Under what conditions does a TFNP class admit a circuit characterization?

A. For every TFNP class there is a model of monotone circuit which characterizes it!

Q. Under what conditions does a monotone circuit model admit a TFNP characterization?

A. Iff the monotone circuit model C has a universal family of functions!

Circuit Complexity

Q. Under what conditions does a TFNP class admit a circuit characterization?

A. For every TFNP class there is a model of monotone circuit which characterizes it!

Q. Under what conditions does a monotone circuit model admit a TFNP characterization?

A. Iff the monotone circuit model C has a universal family of functions!

A monotone function F such that

1. for any partial function g:
   C efficiently computes $g \implies$ there is a string z such that $F \restriction z(x) = g(x)$
   for all x on which g is defined
2. C efficiently computes F

Circuit Complexity

Q. Under what conditions does a TFNP class admit a circuit characterization?

A. For every TFNP class there is a model of monotone circuit which characterizes it!

Q. Under what conditions does a monotone circuit model admit a TFNP characterization?

A. Iff the monotone circuit model C has a universal family of functions! (And closed under low-depth formula reductions).

A monotone function F such that

1. for any partial function g:
   C efficiently computes $g \implies$ there is a string z such that $F \restriction z(x) = g(x)$ for all x on which g is defined
2. C efficiently computes F

Open Problem

Q. A generic lifting theorem?

A circuit and proof system characterization of a TFNP class immediately implies an interpolation theorem. Does the same hold for lifting theorems?