Komplexität von Suchproblemen und Beweissystemen

Ein Abschlusskolloquium von Anton Ehrmanntraut

Welches dieser Suchprobleme ist effizient (ließ: in P-Zeit) lösbar?

Gegeben	**Gesucht**
Graph G	Matching größter Kardinalität in G
Aussagenlog. Formel ψ	Belegung, welche ψ erfüllt (oder „ψ unerfüllbar ausgeben“)
Graph G, $k \in \mathbb{N}$	Clique C in G mit $\geq k$ Knoten (oder „gibt keine Clique d. Größe k“ ausgeben)
Graph G	Clique C in G größter Kardinalität
Graph G	Hamiltonzyklus P in G (oder „nicht hamiltonisch“ ausgeben)
Graph G, Hamiltonzyklus P	Hamiltonzyklus $P' \neq P$ in G (oder „gibt keinen anderen“ ausgeben)
Graphen G, H	Graphisomorphismus σ von G nach H (oder „nicht isomorph“ ausgeben)
Natürliche Zahl $k > 1$	Primfaktor von k

State of the art

Empirische Evidenz, oder *strategic ambiguity*

In der Praxis interessiert man sich vor allem für die Lösung der Optimierungsvariante.

Es ist leicht zu sehen, dass man die Entscheidungsvariante auf die Berechnungsvariante und die Berechnungsvariante auf die Optimierungsvariante zurückführen kann ($E \leq B \leq O$).

Umgekehrt kann man aber auch die Optimierungsvariante auf die Entscheidungsvariante zurückführen ($O \leq E$). Mit Hilfe binärer Suche und durch das Lösen der Entscheidungsvarianten mehrerer Teilprobleme lässt sich eine optimale Rundreise in polynomieller Zeit ermitteln.

Die drei Problemvarianten sind also gleich schwierig ($E \equiv B \equiv O$).

Dies ist typisch für Optimierungsprobleme.

$\Rightarrow$ **es genügt, die Komplexität der Entscheidungsvarianten zu untersuchen**

Empirische Evidenz, oder *strategic ambiguity*

In der Praxis interessiert man sich vor allem für die Lösung der Optimierungsvariante.

Es ist leicht zu sehen, dass man die Entscheidungsvariante auf die Berechnungsvariante und die Berechnungsvariante auf die Optimierungsvariante zurückführen kann ($E \leq B \leq O$).

Umgekehrt kann man aber auch die Optimierungsvariante auf die Entscheidungsvariante zurückführen ($O \leq E$). Mit Hilfe binärer Suche und durch das Lösen der Entscheidungsvarianten mehrerer Teilprobleme lässt sich eine optimale Rundreise in polynomieller Zeit ermitteln.

Die drei Problemvarianten sind also gleich schwierig ($E \equiv B \equiv O$).

Dies ist typisch für Optimierungsprobleme.

$\Rightarrow$ **es genügt, die Komplexität der Entscheidungsvarianten zu untersuchen**

Empirische Evidenz, oder *strategic ambiguity*

In der Praxis interessiert man sich vor allem für die Lösung der Optimierungsvariante.

Es ist leicht zu sehen, dass man die Entscheidungsvariante auf die Berechnungsvariante und die Berechnungsvariante auf die Optimierungsvariante zurückführen kann ($E \leq B \leq O$).

Umgekehrt kann man aber auch die Optimierungsvariante auf die Entscheidungsvariante zurückführen ($O \leq E$). Mit Hilfe binärer Suche und durch das Lösen der Entscheidungsvarianten mehrerer Teilpro
ermitte

Die dre

Dies is

⇒ **es**
zu

Exercise 2 – Decision, optimisation, construction

Strictly speaking, most of complexity theory is about *decision* problems: e.g. does this graph have an independent set of size k. We have been talking mostly about *optimisation* problems: what is the maximum size of an independent set in this graph. In practice, one often cares about the *construction* problem: compute an independent set of maximum size in this graph.

It is well known that for problems in the class NP these three notions are basically (polynomial-time) equivalent. That is, if you can solve one, you can solve the others. But that doesn't mean that there aren't better and worse ways to go about it.

Empirische Evidenz, oder *strategic ambiguity*

In der Praxis interessiert man sich vor allem für die Lösung der Optimierungsvariante.

Es ist leicht zu sehen, dass man die Entscheidungsvariante auf die Berechnungsvariante und die Berechnungsvariante auf die Optimierungsvariante zurückführen kann ($E \leq B \leq O$).

Umgekehrt kann man aber auch die Optimierungsvariante auf die Entscheidungsvariante zurückführen ($O \leq E$). Mit Hilfe binärer Suche und durch das Lösen der Entscheidungsvarianten mehrerer Teilpro

ermitte

Die dre

Dies is

$\Rightarrow$ **es**

zu

Exercise 2 – Decision, optimisation, construction

Strictly speaking, most of complexity theory is about *decision* problems: e.g. does this graph have an independent set of size k. We have been talking mostly about *optimisation* problems: what is the maximum size of an independent set in this graph. In practice, one often cares about the *construction* problem: compute an independent set of maximum size in this graph.

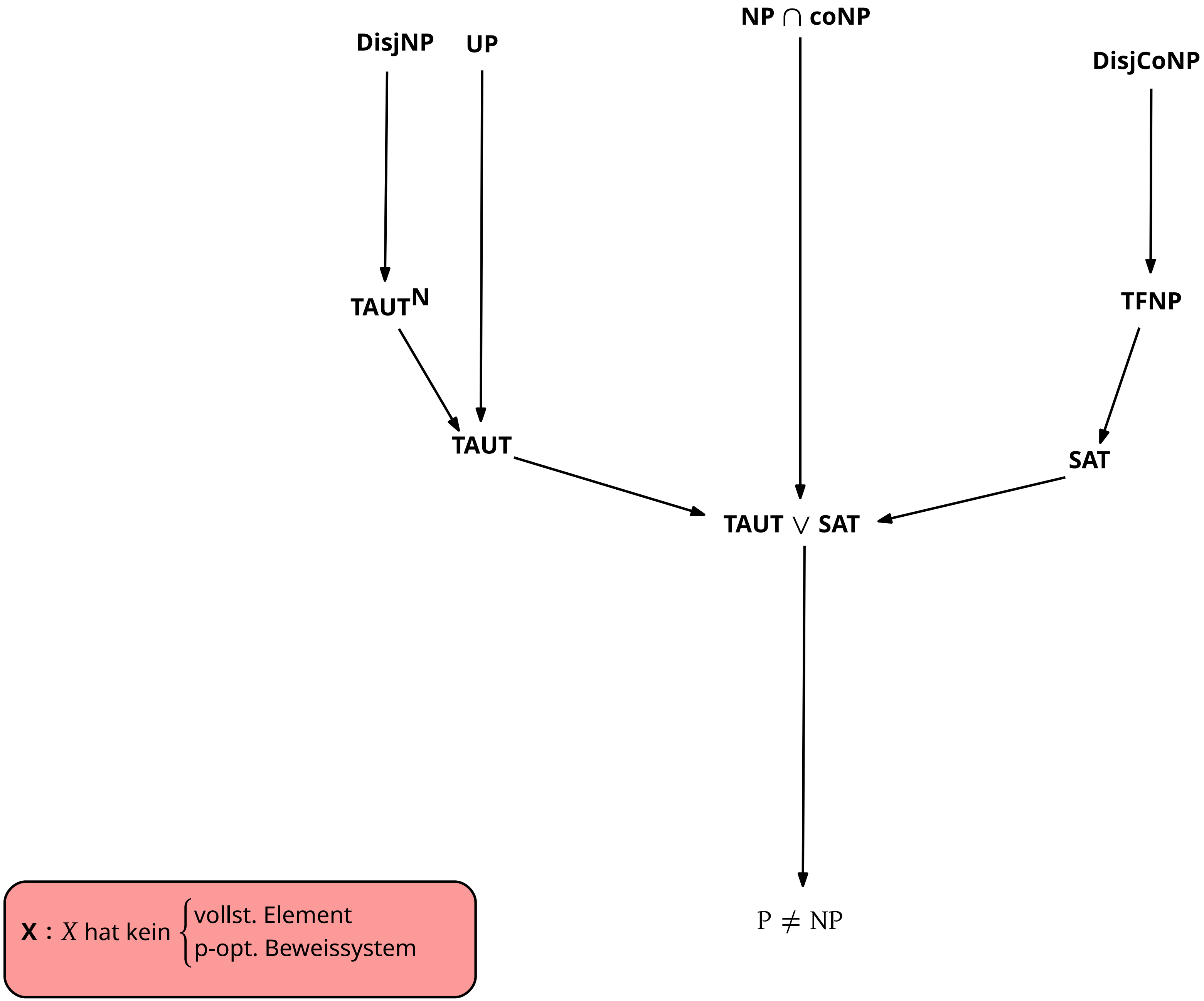
offensichtlich „NP-vollständig“ gemeint!

It is well known that for problems in the class NP these three notions are basically (polynomial-time) equivalent. That is, if you can solve one, you can solve the others. But that doesn't mean that there aren't better and worse ways to go about it.

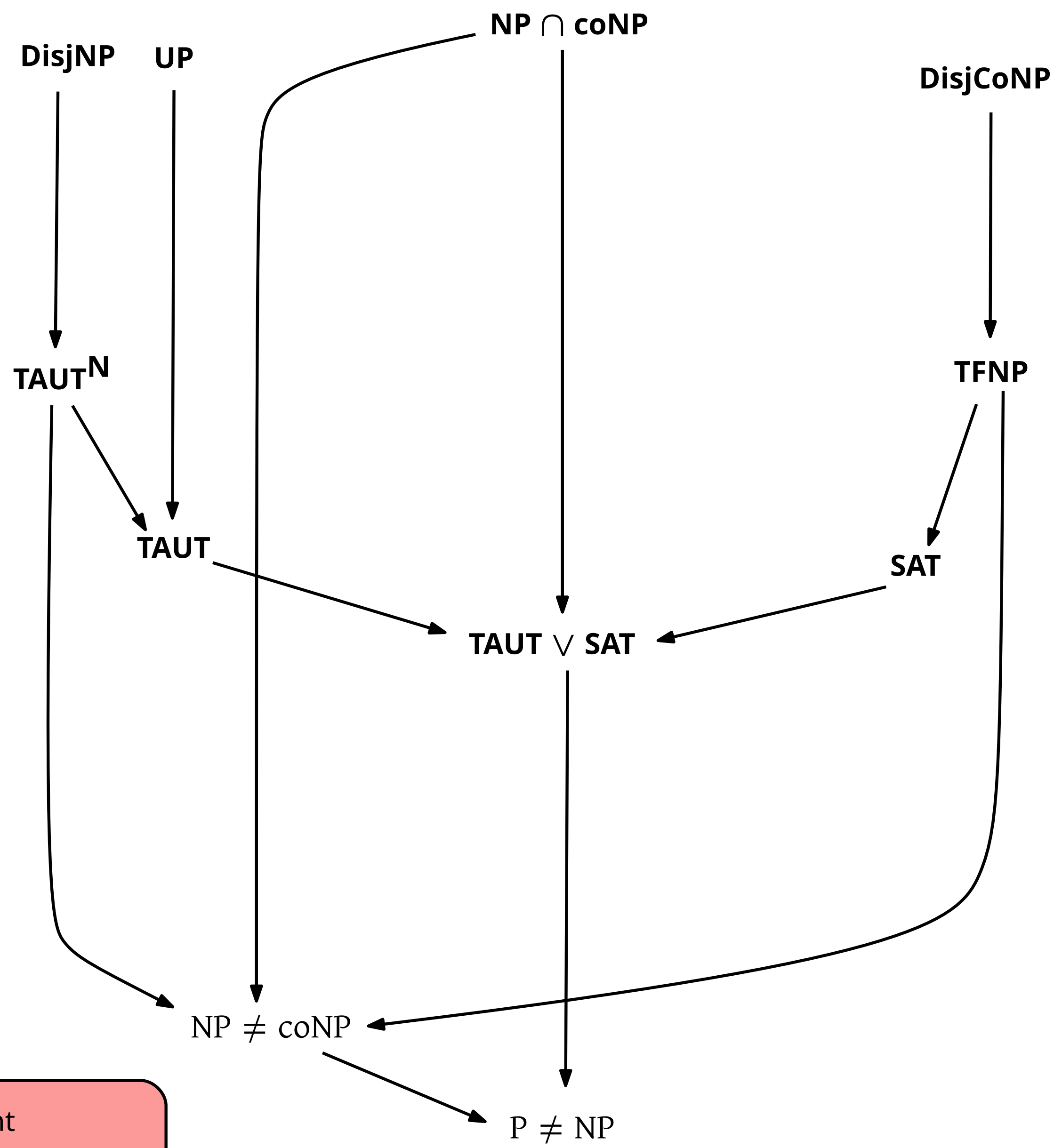
TAUT	:	ex. kein p-optimales Beweissystem für TAUT
TAUT^{N}	:	ex. keine optimales Beweissystem für TAUT
SAT	:	ex. kein p-optimales Beweissystem für SAT
TFNP	:	ex. keine many-one-vollständige NP-Relation für TFNP
NP $\cap$ coNP	:	ex. keine many-one-vollständige Menge für NP $\cap$ coNP
UP	:	ex. keine many-one-vollständige Menge für UP
DisjNP	:	ex. kein many-one-vollständiges Paar für DisjNP
DisjCoNP	:	ex. kein many-one-vollständiges Paar für DisjCoNP

TAUT	:	ex. kein p-optimales Beweissystem für TAUT
TAUT$^{\mathbf{N}}$	:	ex. keine optimales Beweissystem für TAUT
SAT	:	ex. kein p-optimales Beweissystem für SAT
TFNP	:	ex. keine many-one-vollständige NP-Relation für TFNP
NP $\cap$ coNP	:	ex. keine many-one-vollständige Menge für $\mathrm{NP} \cap \mathrm{coNP}$
UP	:	ex. keine many-one-vollständige Menge für UP
DisjNP	:	ex. kein many-one-vollständiges Paar für DisjNP
DisjCoNP	:	ex. kein many-one-vollständiges Paar für $\mathrm{DisjCoNP}$

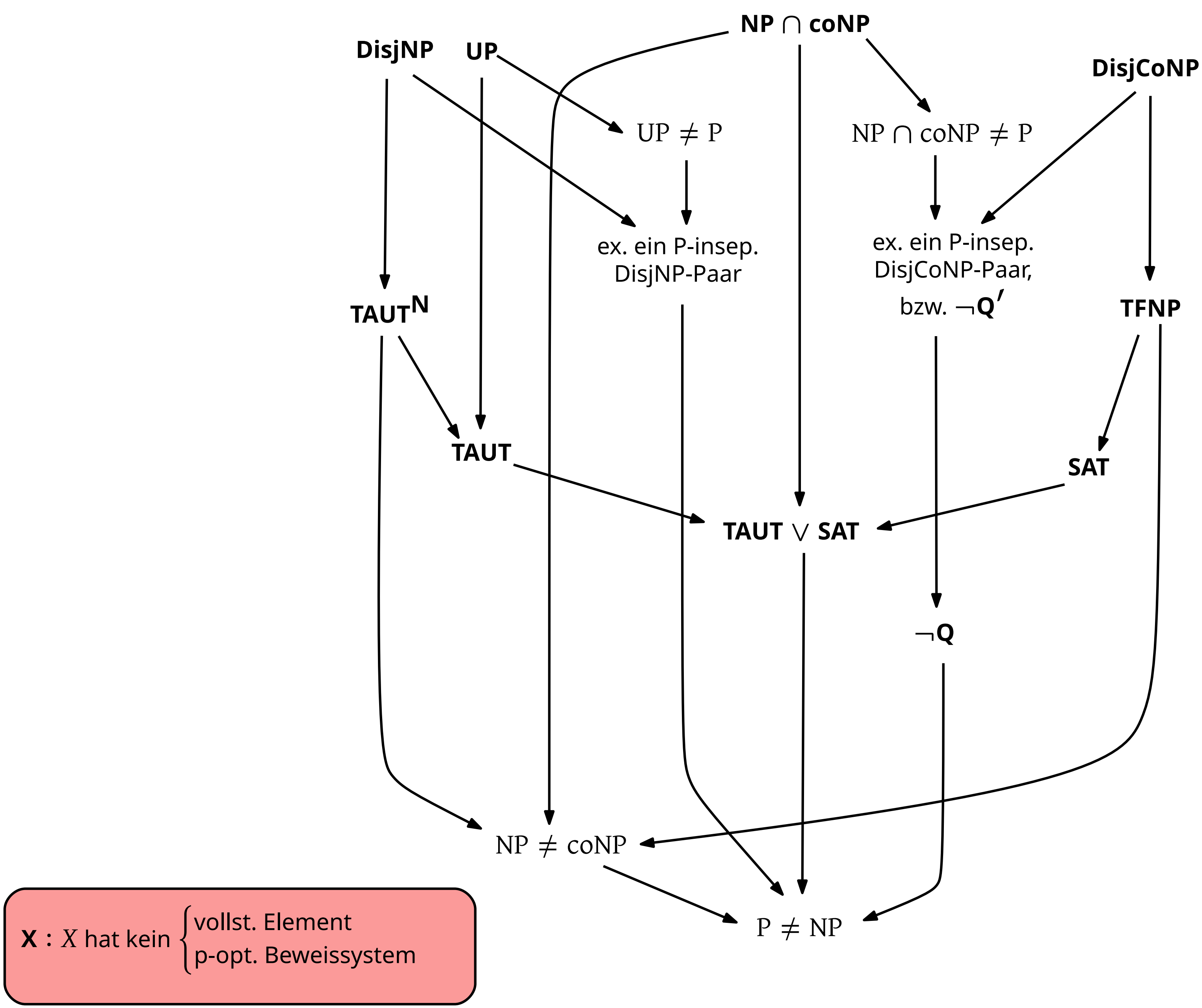
Konvention: $\mathbf{X} \triangleq X$ hat kein $\begin{cases} \text{vollst. Element} \\ \text{p-opt. Beweissystem} \end{cases}$



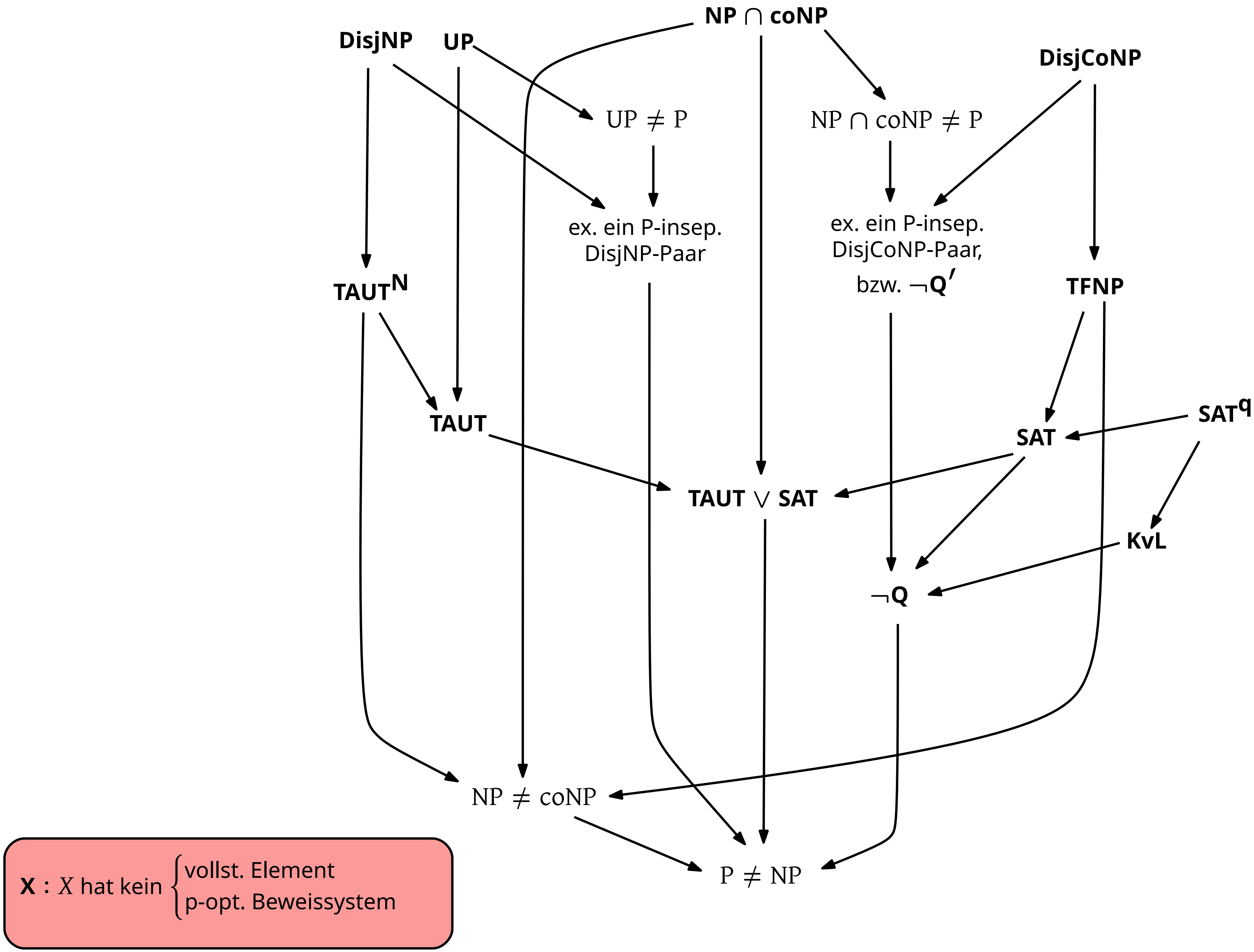
DisjNP
UP
NP ∩ coNP
DisjCoNP
TAUT^{N}
TFNP
TAUT
SAT
TAUT ∨ SAT
P ≠ NP
X : X hat kein
vollst. Element
p-opt. Beweissystem



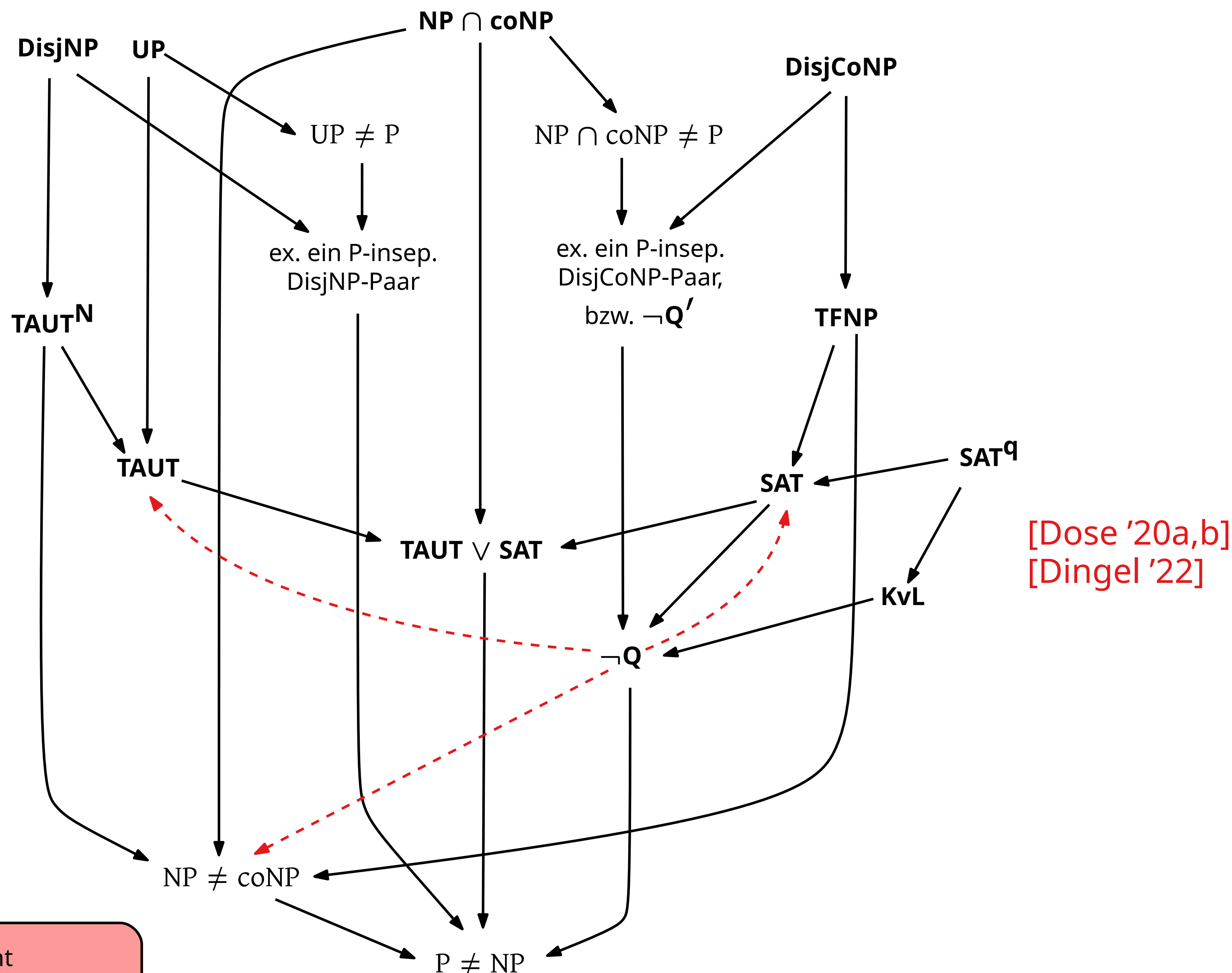
X : X hat kein $\begin{cases}\text{vollst. Element}\\ \text{p-opt. Beweissystem}\end{cases}$



NP ∩ coNP
DisjNP
UP
DisjCoNP
UP ≠ P
NP ∩ coNP ≠ P
ex. ein P-insep. DisjNP-Paar
ex. ein P-insep. DisjCoNP-Paar, bzw. ¬**Q′**
TAUT^N
TFNP
TAUT
SAT
TAUT ∨ SAT
¬**Q**
NP ≠ coNP
P ≠ NP
X : *X* hat kein {vollst. Element / p-opt. Beweissystem}

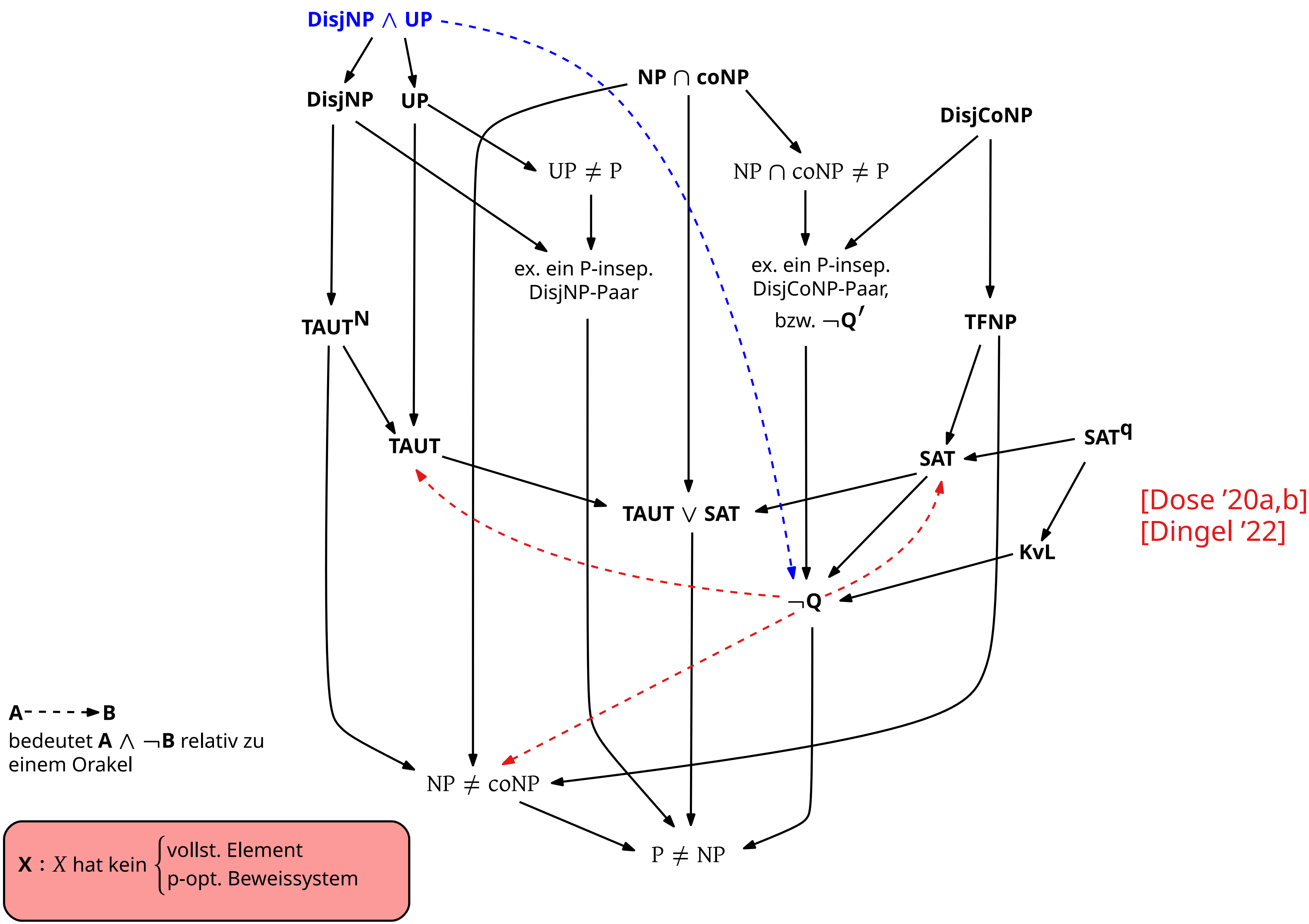


NP ∩ coNP
DisjNP
UP
DisjCoNP
UP ≠ P
NP ∩ coNP ≠ P
ex. ein P-insep.
DisjNP-Paar
ex. ein P-insep.
DisjCoNP-Paar,
bzw. ¬Q′
TAUT^N
TFNP
TAUT
SAT^q
SAT
TAUT ∨ SAT
KvL
¬Q
NP ≠ coNP
P ≠ NP
X : X hat kein
vollst. Element
p-opt. Beweissystem

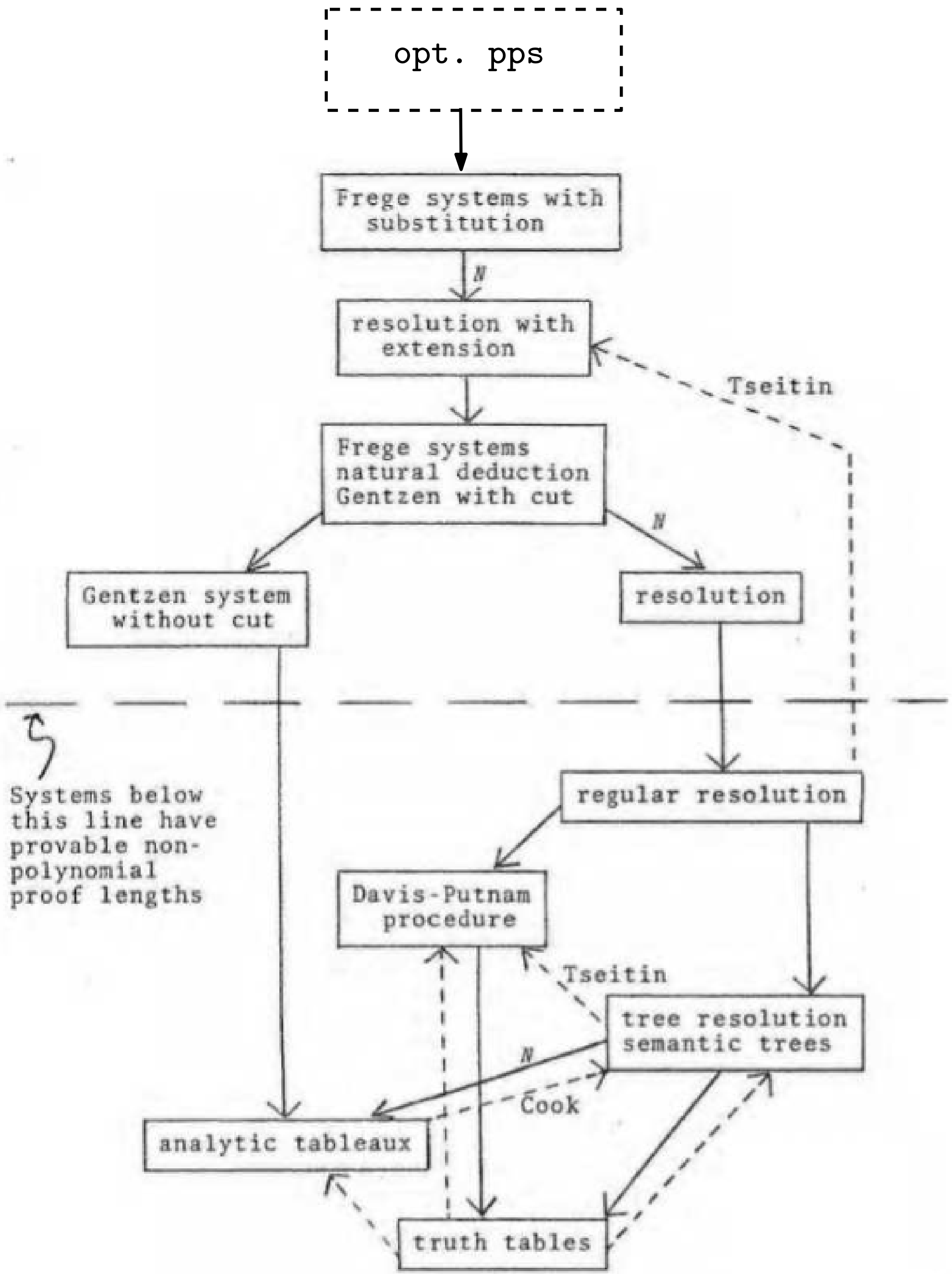


A - - - -> **B**
bedeutet **A** $\wedge$ $\neg$**B** relativ zu einem Orakel

X : X hat kein $\begin{cases} \text{vollst. Element} \\ \text{p-opt. Beweissystem} \end{cases}$


DisjNP ∧ UP
DisjNP
UP
NP ∩ coNP
DisjCoNP
UP ≠ P
NP ∩ coNP ≠ P
ex. ein P-insep. DisjNP-Paar
ex. ein P-insep. DisjCoNP-Paar, bzw. ¬Q′
TAUT^N
TFNP
TAUT
SAT
SAT^q
TAUT ∨ SAT
KvL
¬Q
[Dose ’20a,b]
[Dingel ’22]
NP ≠ coNP
P ≠ NP
A - - - → B
bedeutet A ∧ ¬B relativ zu einem Orakel
X : X hat kein vollst. Element
p-opt. Beweissystem

[Buss '21]

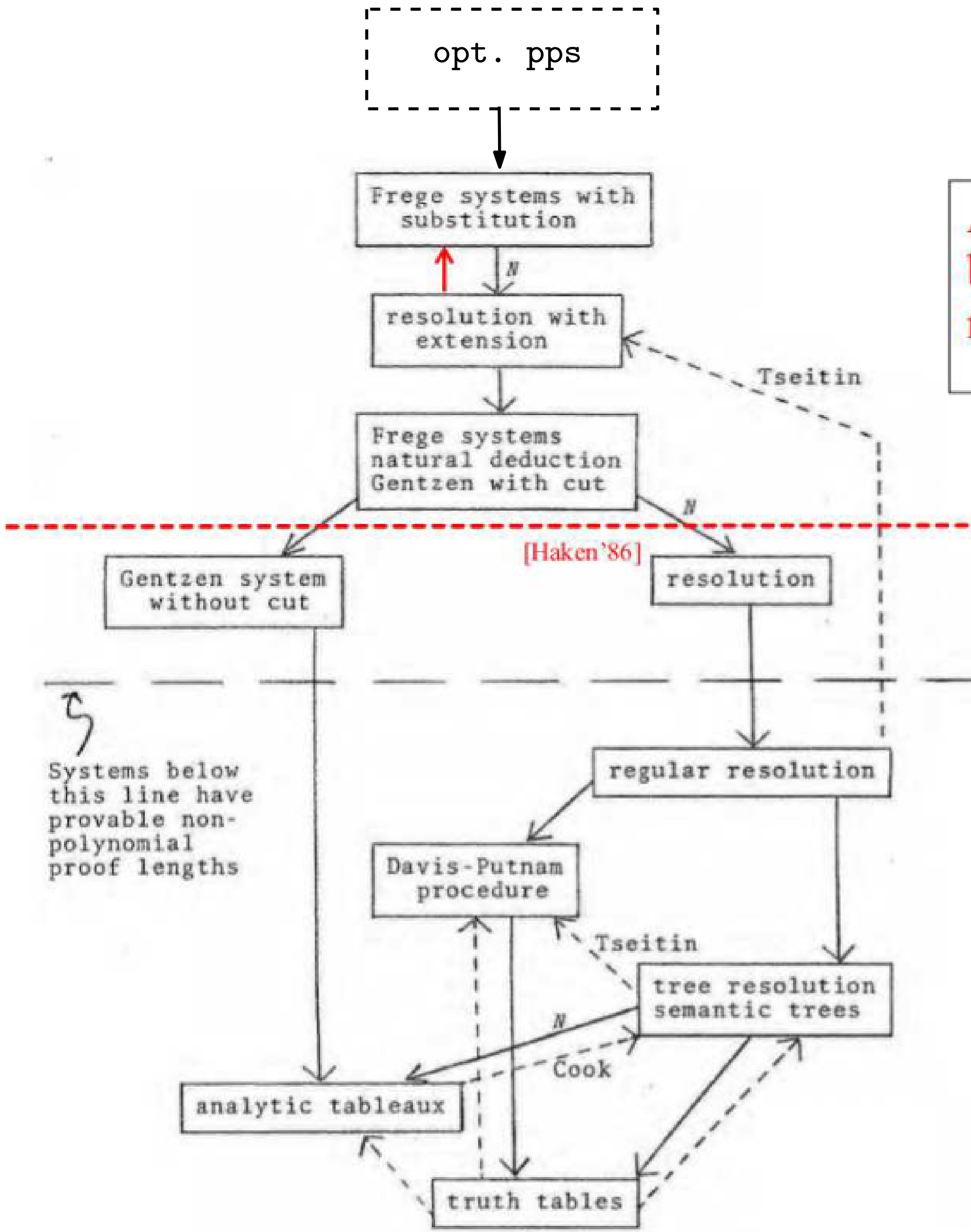


R.A. Reckhow, PhD thesis, 1975

Cook's Program: Prove NP≠coNP by proving there is no polynomially bounded propositional proof system.

As of 1975: Systems above the line were not known to not be polynomially bounded.

[Buss ’21]



R.A. Reckhow, PhD thesis, 1975

As of 2015, proof systems below the line are known to not be polynomially bounded:

Constant-depth (AC^0) Frege
[Ajtai’88; Pitassi-Beame-Impagliazzo’93; Krajicek-Pudlak-Woods’95]

Constant-depth Frege with counting mod *m* axioms
[Ajtai’94; Beame-Impagliazzo-Krajicek-Pitassi-Pudlak’96; B-Impagliazzo-Krajicek-Pudlak-Razborov-Sgall’96; Grigoriev’98]

Cutting Planes
[Pudlak’97]

Nullstellensatz
[B-Impagliazzo-Krajicek-Pudlak-Razborov-Sgall’96; Grigoriev’98]

Polynomial calculus
[Razborov’98; Impagliazzo-Pudlak-Sgall’99; Ben-Sasson-Impagliazzo’99; B-Grigoriev-Impagliazzo-Pitassi’96; B-Impagliazzo-Krajicek-Pudlak-Razborov-Sgall’96; Alekhnovich-Razborov’01]