A path to compute the 9th Dedekind Number using FPGA Supercomputing

Lennart Van Hirtum

Patrick De Causmaecker – Promotor

Jens Goemaere – Mentor

Achtergrond

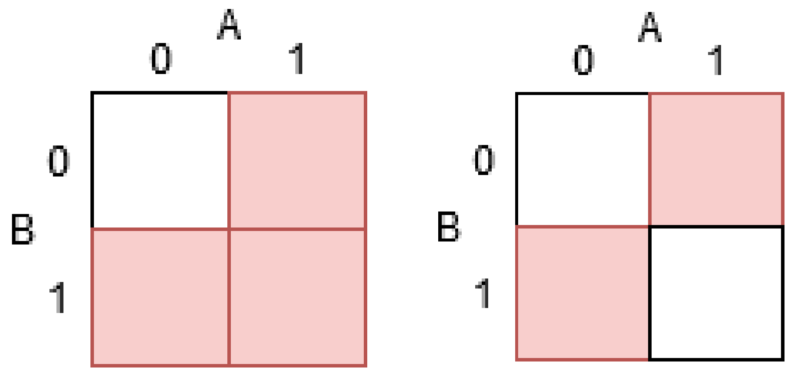
- N-de Dedekind Getal:

  → “Het aantal monotoon booleaanse functies met n variabelen”

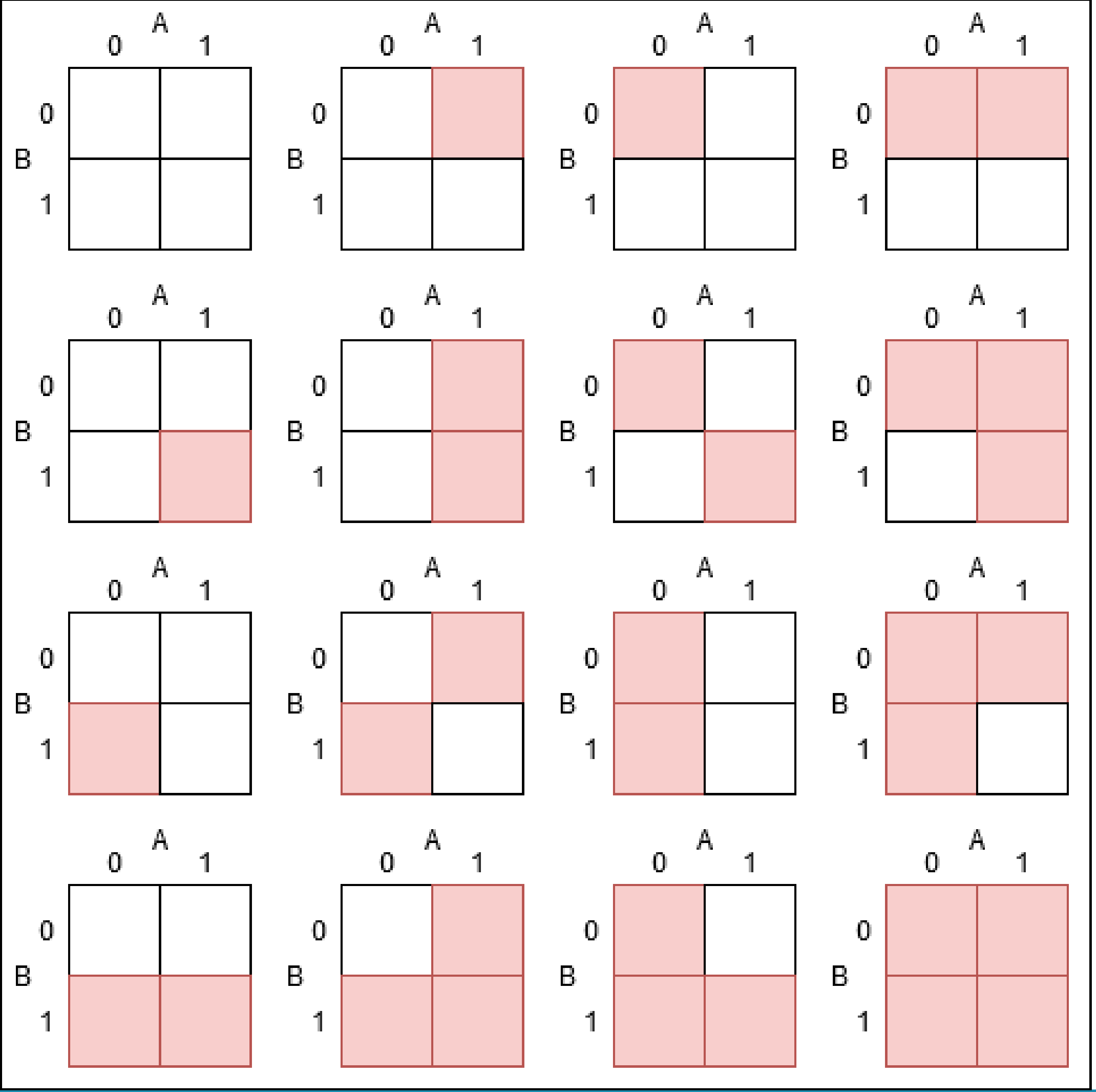
- Monotoon:

  → Keurige splitsing tussen de 0-waarden en 1-waarden van de functie

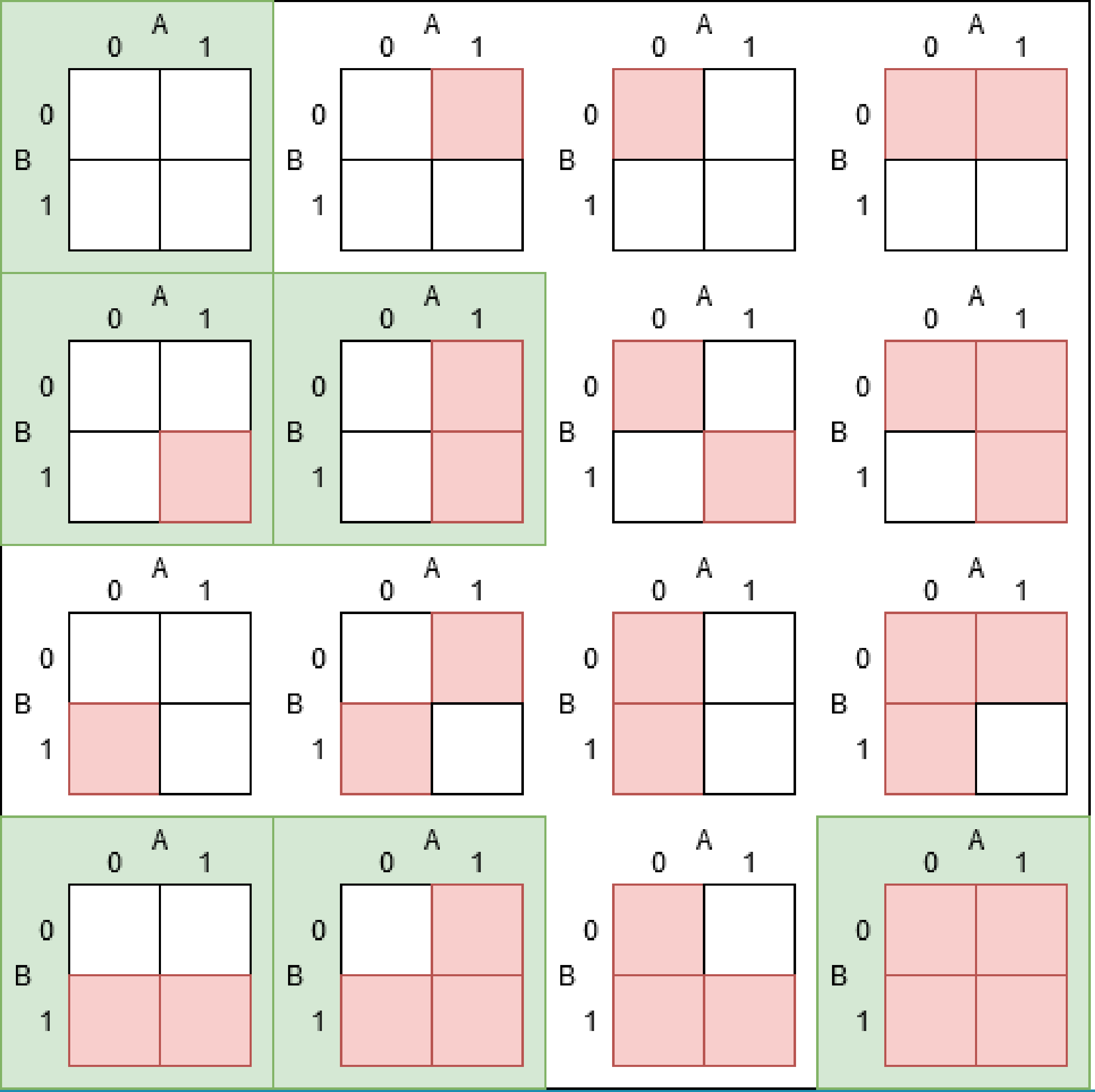
  → AND/OR zijn monotoon
  XOR niet



KU LEUVEN



KU LEUVEN



$$D(2) = 6$$

Gekende Dedekind Getallen

#Monotoon Booleaanse Functies in N variabelen

$D(0) = 2$	Dedekind (1897)
$D(1) = 3$	Dedekind (1897)
$D(2) = 6$	Dedekind (1897)
$D(3) = 20$	Dedekind (1897)
$D(4) = 168$	Dedekind (1897)
$D(5) = 7581$	Church (1940)
$D(6) = 7828354$	Ward (1946)
$D(7) = 2414682040998$	Church (1965)
$D(8) = 56130437228687557907788$	Wiedemann (1991)
$D(9) = ?$	

Doug Wiedemann. A computation of the eighth dedekind number. Springer Link 10.1007 1991

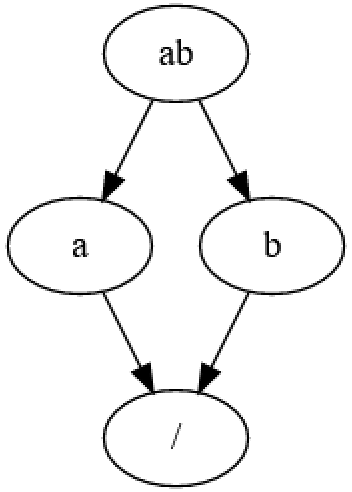
Complexiteit van het probleem

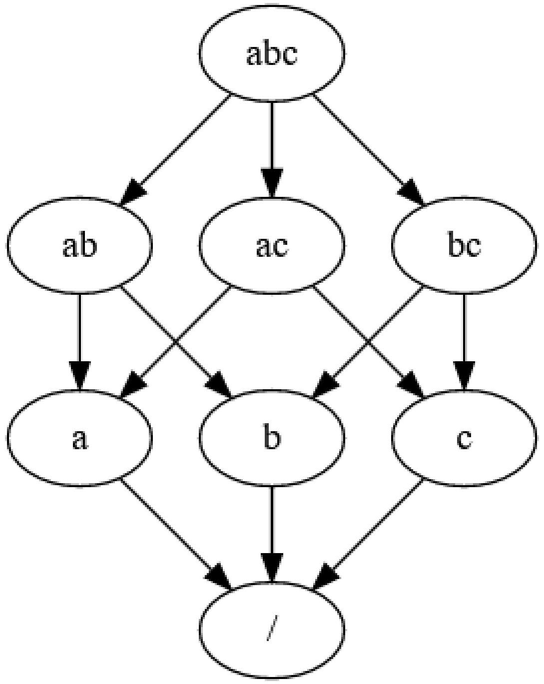
$$O\left(2^{2^n}\right)$$

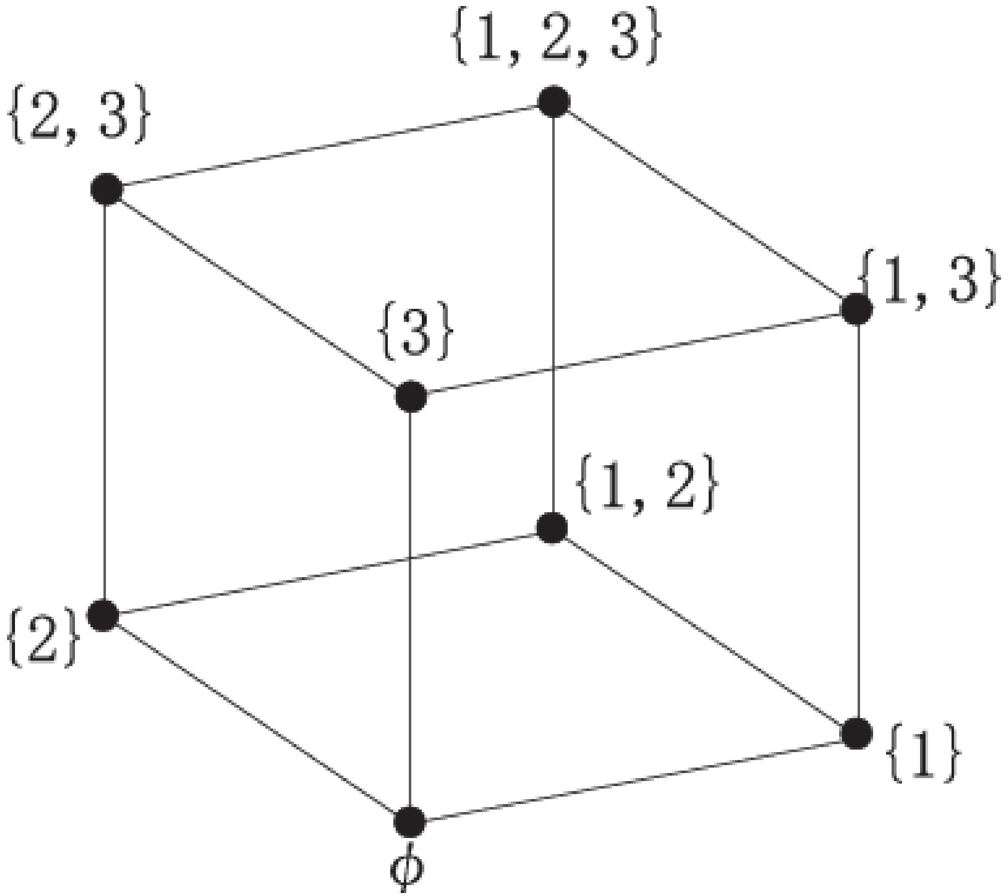
- Naïeve methode, alle Booleaanse functies afgaan en monotone tellen
- Vermoeden dat het niet beter kan dan dubbele exponentieel
- Grote factoren wel te winnen

Hyperkubus Visualisatie

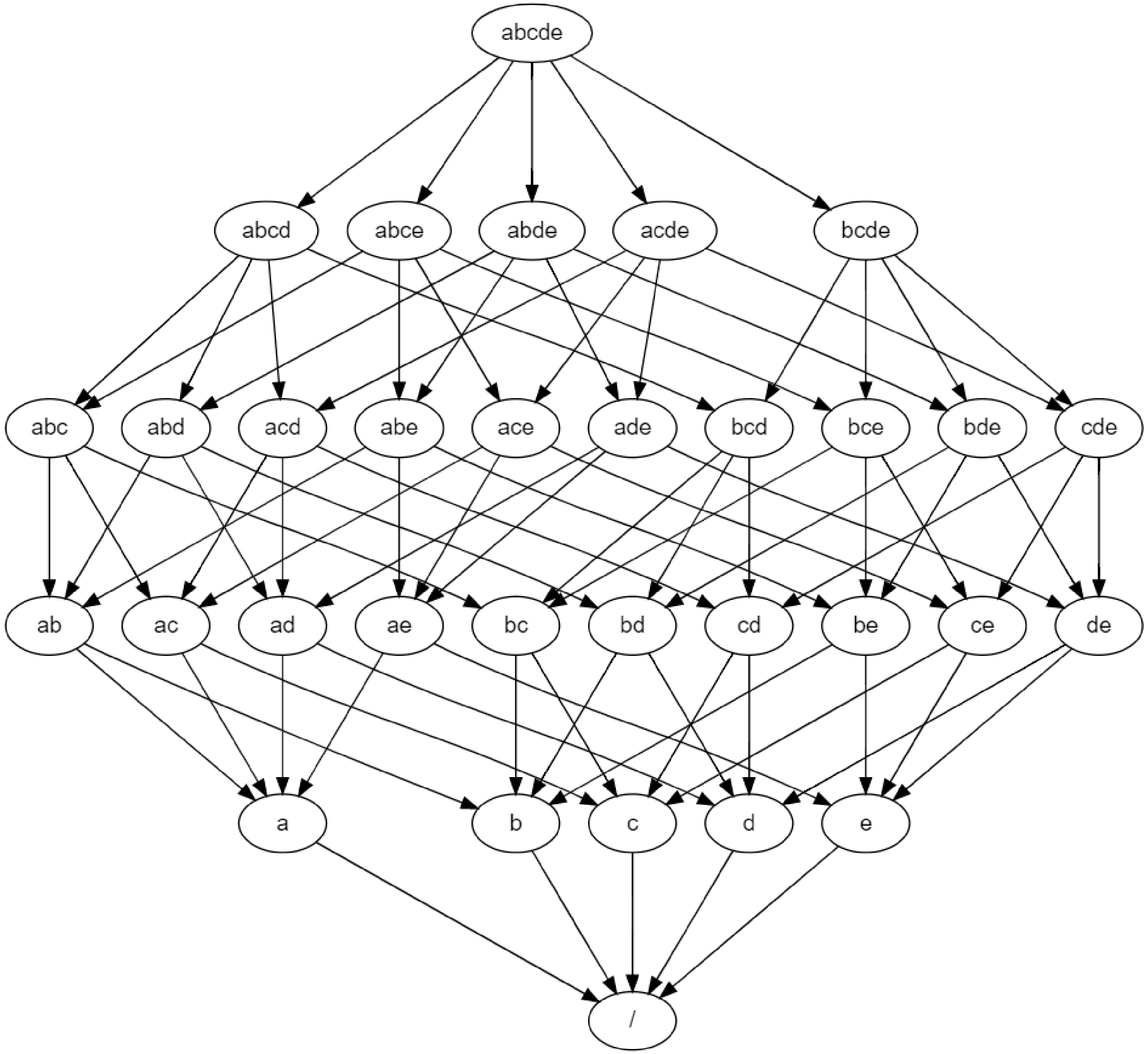
- Nodes opgedeeld in lagen, per aantal geactiveerde variabelen
- Nodes kunnen gekleurd worden
  adhv return 0 of 1





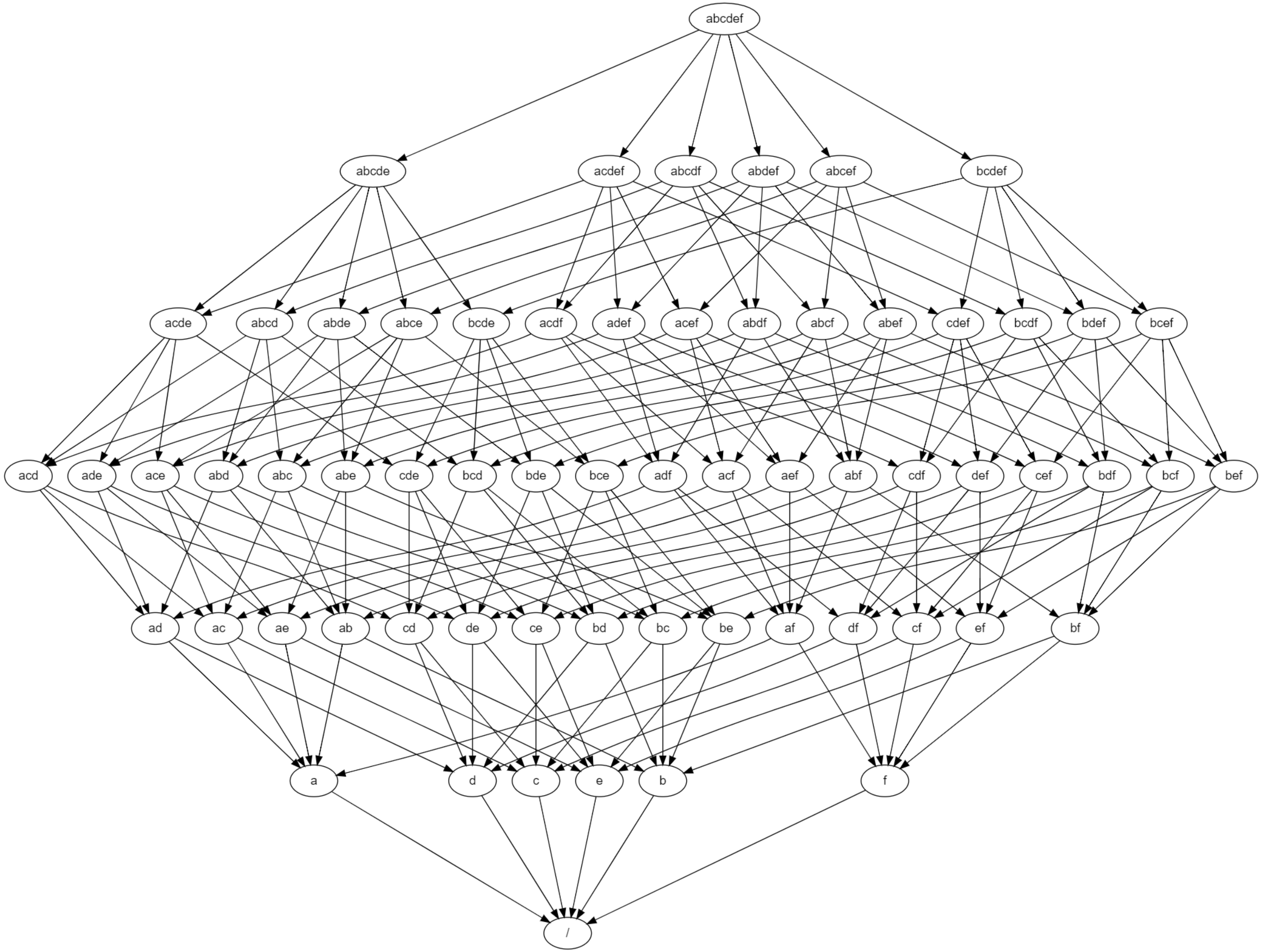


5D

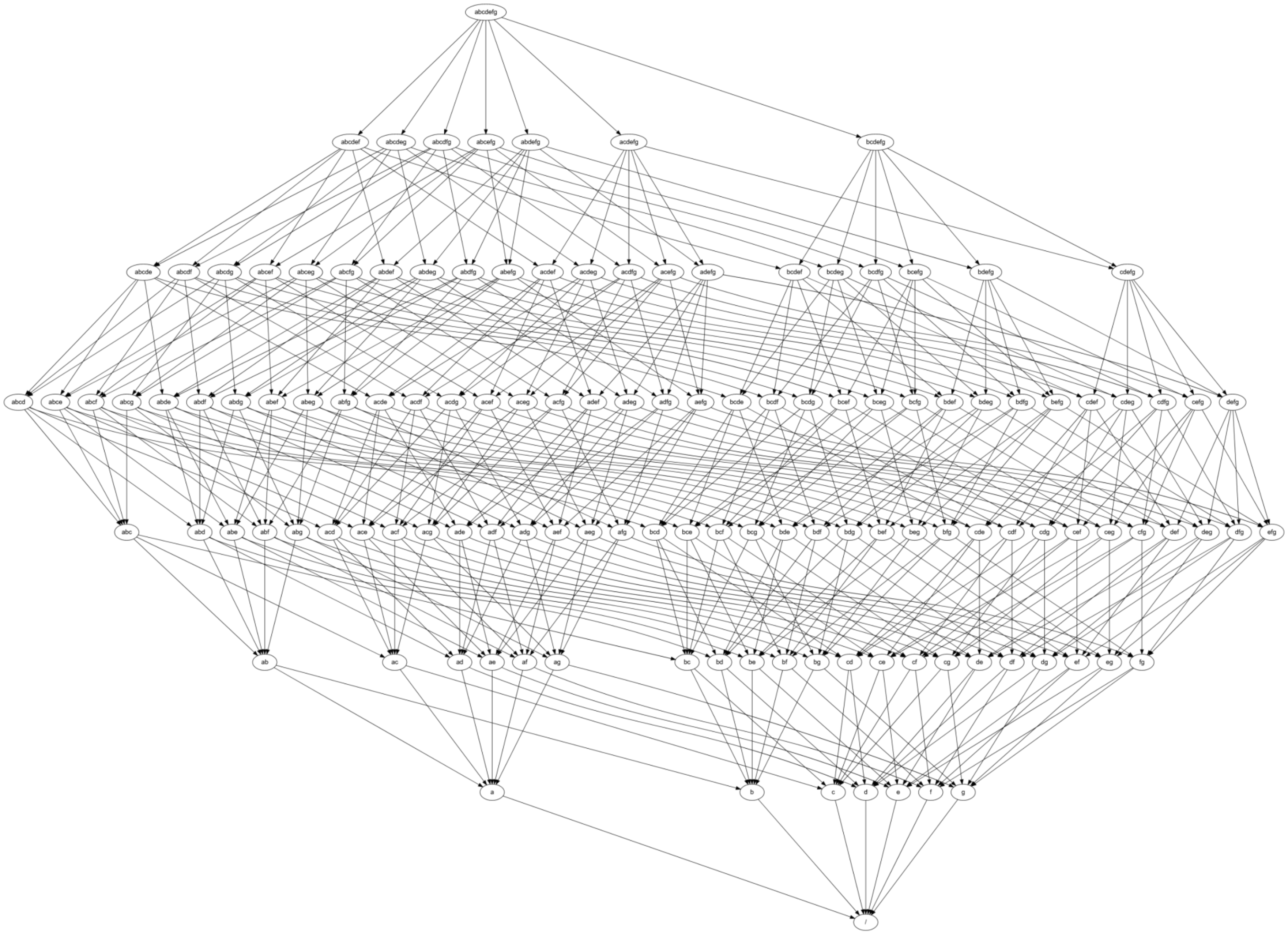

abcde
abcd
abce
abde
acde
bcde
abc
abd
acd
abe
ace
ade
bcd
bce
bde
cde
ab
ac
ad
ae
bc
bd
cd
be
ce
de
a
b
c
d
e
/

KU LEUVEN

6D

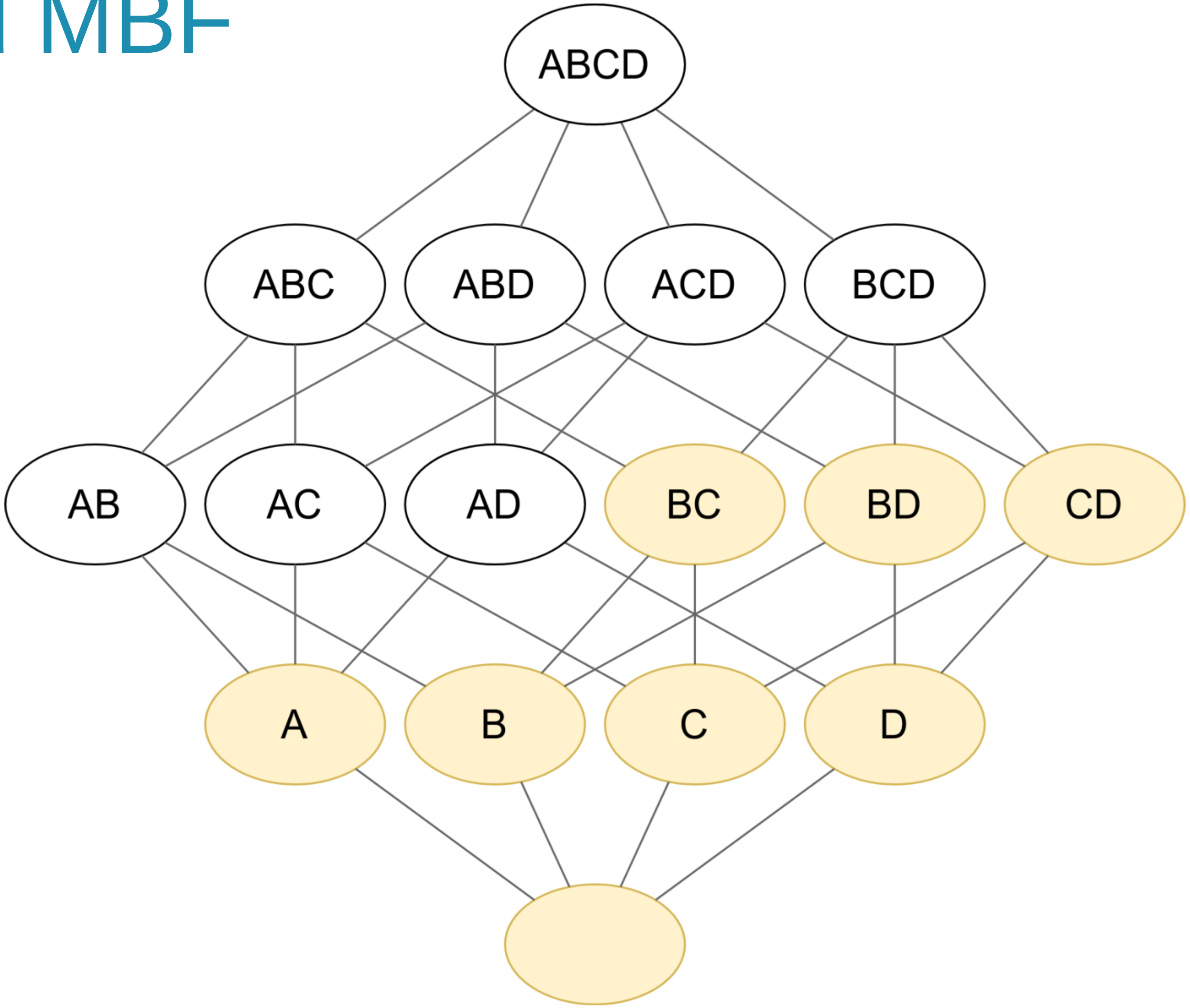

abcdef
abcde
acdef
abcdf
abdef
abcef
bcdef
acde
abcd
abde
abce
bcde
acdf
adef
acef
abdf
abcf
abef
cdef
bcdf
bdef
bcef
acd
ade
ace
abd
abc
abe
cde
bcd
bde
bce
adf
acf
aef
abf
cdf
def
cef
bdf
bcf
bef
ad
ac
ae
ab
cd
de
ce
bd
bc
be
af
df
cf
ef
bf
a
d
c
e
b
f
/

7D


abcdefg
abcdef
abcdeg
abcdfg
abcefg
abdefg
acdefg
bcdefg
abcde
abcdf
abcdg
abcef
abceg
abcfg
abdef
abdeg
abdfg
abefg
acdef
acdeg
acdfg
acefg
adefg
bcdef
bcdeg
bcdfg
bcefg
bdefg
cdefg
abcd
abce
abcf
abcg
abde
abdf
abdg
abef
abeg
abfg
acde
acdf
acdg
acef
aceg
acfg
adef
adeg
adfg
aefg
bcde
bcdf
bcdg
bcef
bceg
bcfg
bdef
bdeg
bdfg
befg
cdef
cdeg
cdfg
cefg
defg
abc
abd
abe
abf
abg
acd
ace
acf
acg
ade
adf
adg
aef
aeg
afg
bcd
bce
bcf
bcg
bde
bdf
bdg
bef
beg
bfg
cde
cdf
cdg
cef
ceg
cfg
def
deg
dfg
efg
ab
ac
ad
ae
af
ag
bc
bd
be
bf
bg
cd
ce
cf
cg
de
df
dg
ef
eg
fg
a
b
c
d
e
f
g
/

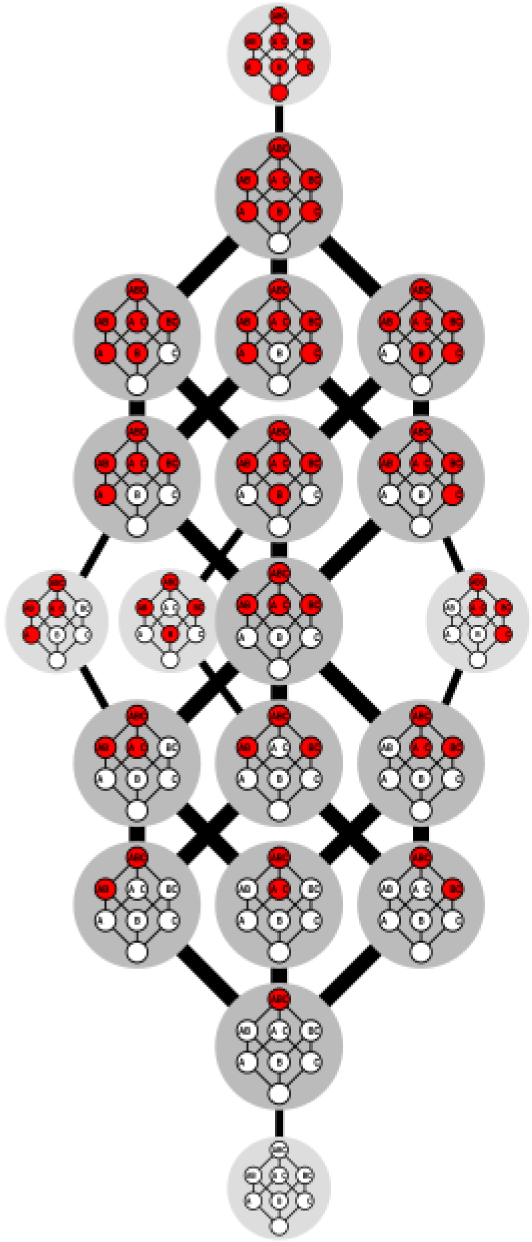
KU LEUVEN

Voorbeeld MBF



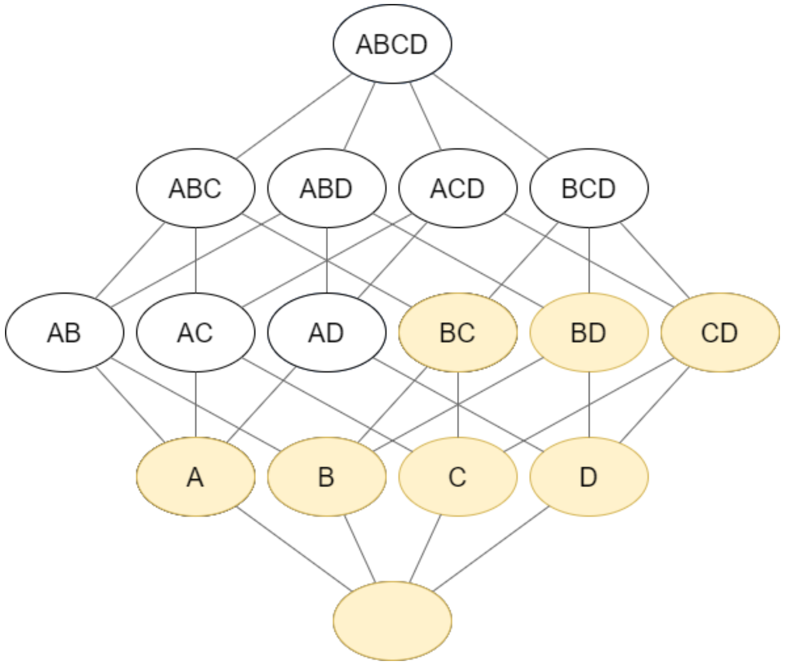
Partiële orde en Intervallen

- $\alpha \leq \beta \Leftrightarrow \alpha$ is een subset van β
- $\gamma \in [\alpha, \beta] \leftrightarrow \alpha \leq \gamma \leq \beta$
- Intervalgrootte $|[\alpha, \beta]|$

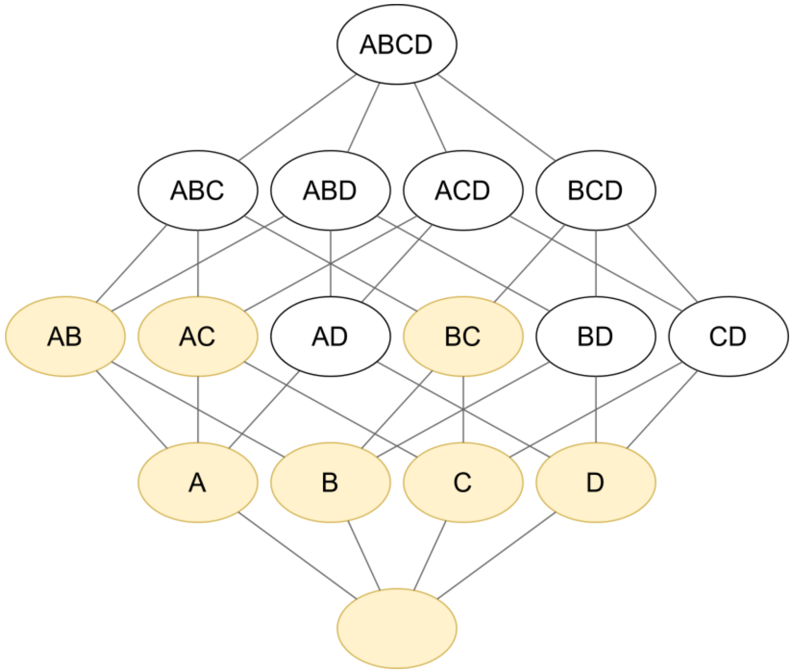


Equivalente MBFs

- Equivalent over hernoeming van de variabelen
- Behoud van eigenschappen

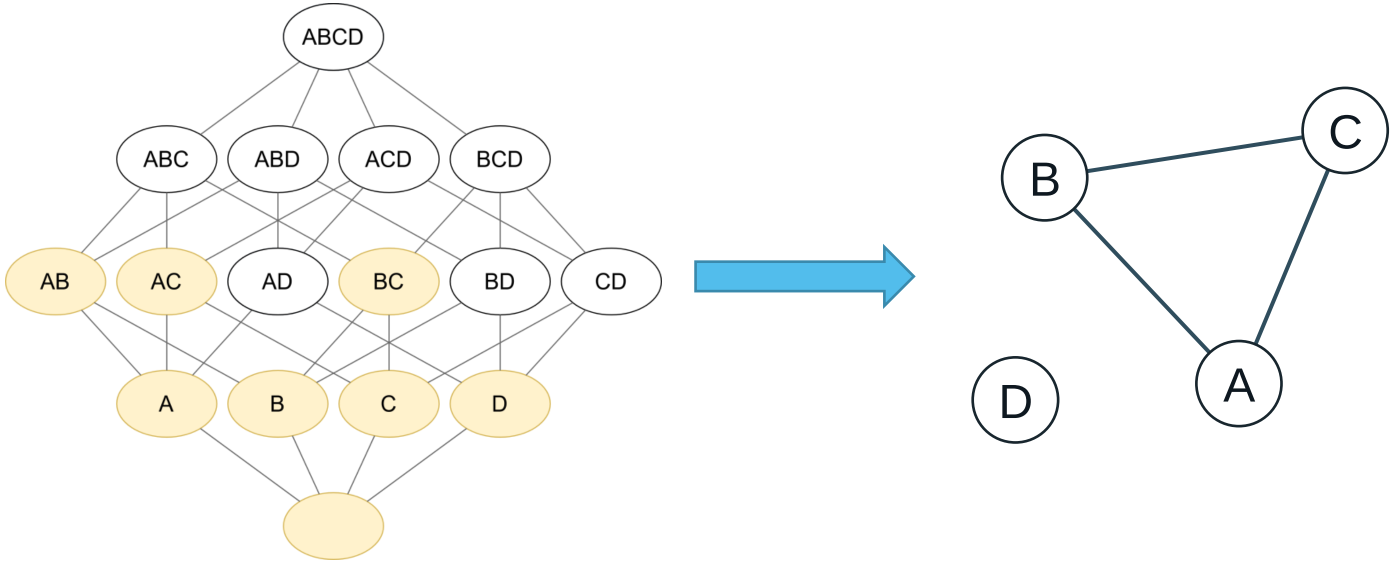


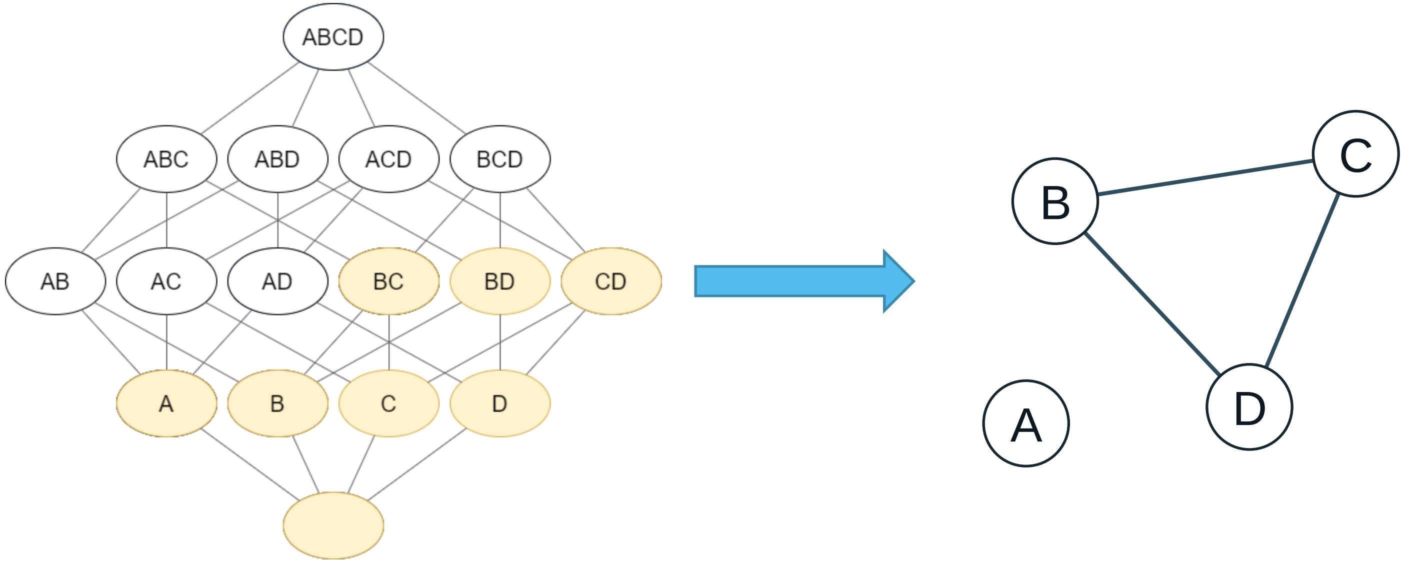
$\cong$



Equivalentieklassen

- Elk met een representant
- Voorstelling als hypergrafe
- Canonisatie





KU LEUVEN

Aantal Equivalentieklassen: $R(N)$

#Equivalentieklassen

$R(0) = 2$
$R(1) = 3$
$R(2) = 5$
$R(3) = 10$
$R(4) = 30$
$R(5) = 210$
$R(6) = 16353$
$R(7) = 490013148$
$R(8) = ?$

Tamon Stephen and Timothy Yusun. Counting inequivalent monotone boolean functions, 2012.

Pad naar $D(9)$

- Efficiënte representatie en operaties
  → Canonisatie
- Vind alle inequivalente MBFs in 7 variabelen ⇒ $(R(7) = 490\,013\,148)$
- Intervalgroottes berekenen $\forall \alpha \in R_7$ bereken $|[\perp, \alpha]|$
  → Bereken $D(8)$ om te verifiëren
- P-Coefficiënten methode
  → $D(8)$ Benchmarks
- FPGA Implementatie voor $D(9)$

Pad naar $D(9)$

- Efficiënte representatie en operaties
  - → Canonisatie
- Vind alle inequivalente MBFs in 7 variabelen $\Rightarrow$ $(R(7) = 490\,013\,148)$
- Intervalgroottes berekenen $\forall \alpha \in R_7$ bereken $|[\bot, \alpha]|$
  - → Bereken $D(8)$ om te verifiëren
- P-Coefficiënten methode
  - → $D(8)$ Benchmarks
- FPGA Implementatie voor $D(9)$

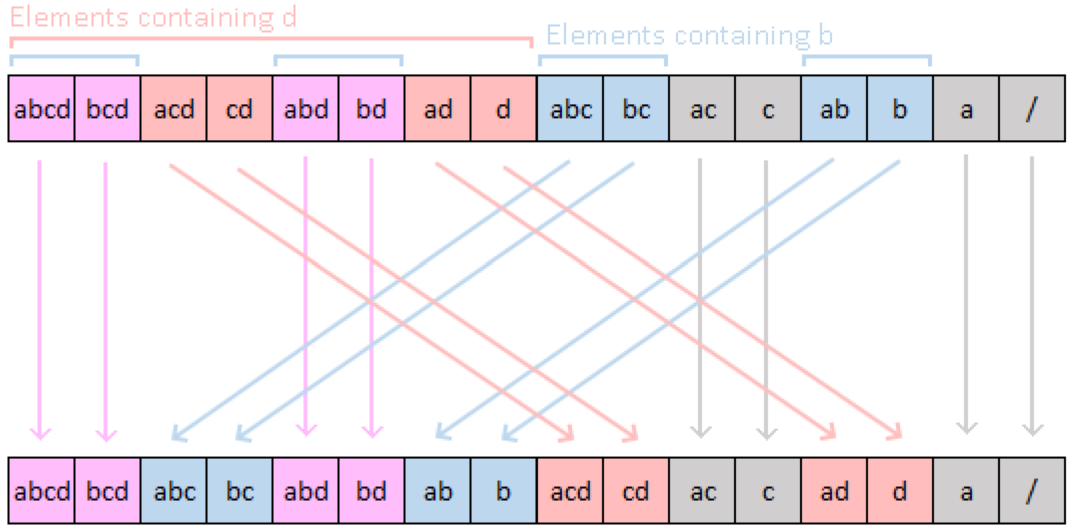
Efficiënte Representatie

- Representeer MBFs als bitset
- Binaire positie van de bit geeft geactiveerde variabelen weer

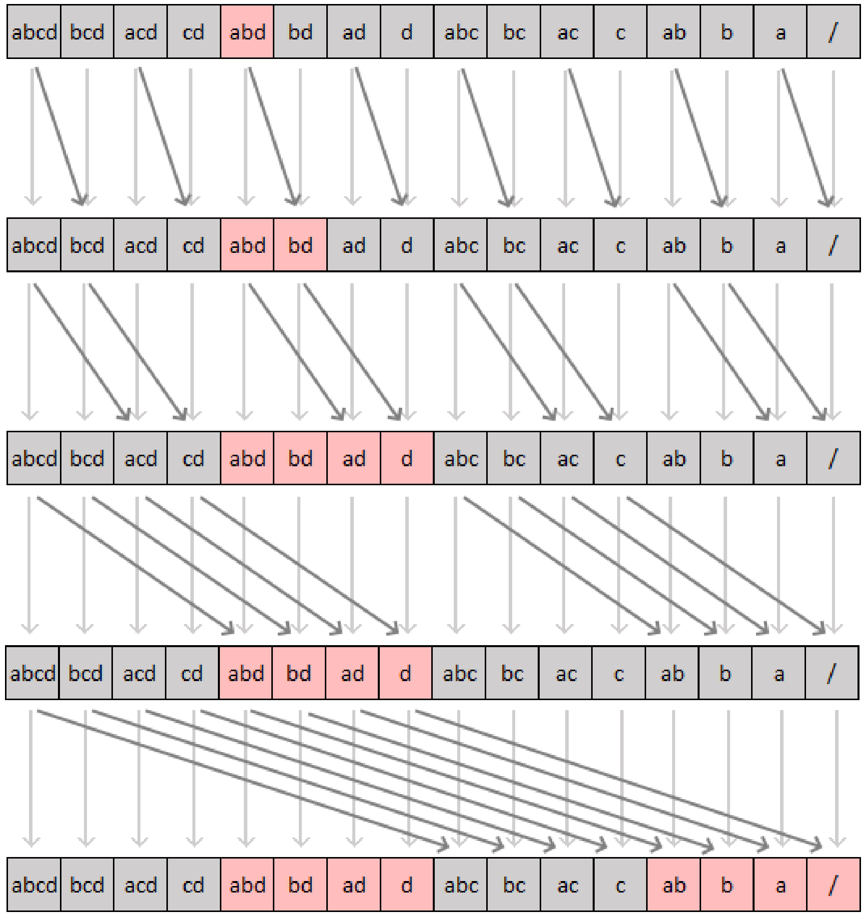
0	1	0	1	1	1	1	1
abc	bc	ac	c	ab	b	a	{}
111	110	101	100	011	010	001	000

Efficiënte Operaties

- Booleaanse operatoren
- shifts



varSwap

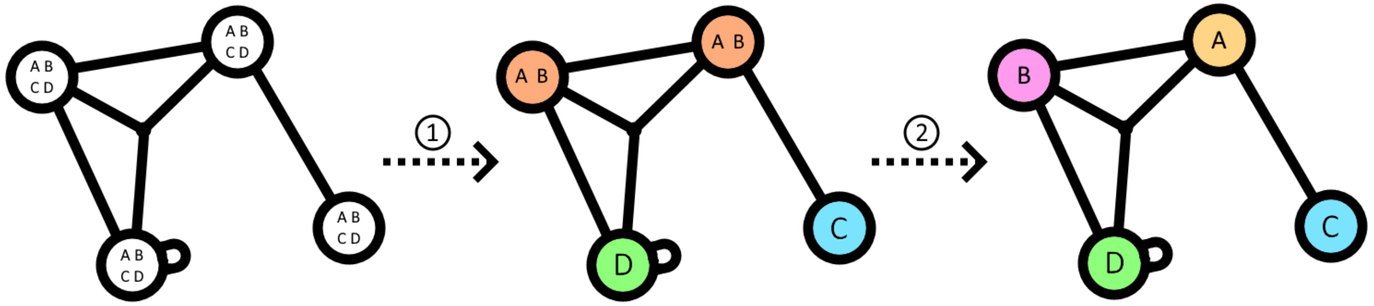


monotonizeDown

KU LEUVEN

Efficiënte Canonisatie

- Zet een MBF om naar de representant van zijn equivalentieklasse
- Naïeve methode:
  - → Itereer over alle permutaties
  - → Vind min/max van arbitraire totale orderelatie
- Verbeteringen
  - → Groeperen van knopen
  - → Iteratief groepen verfijnen



KU LEUVEN

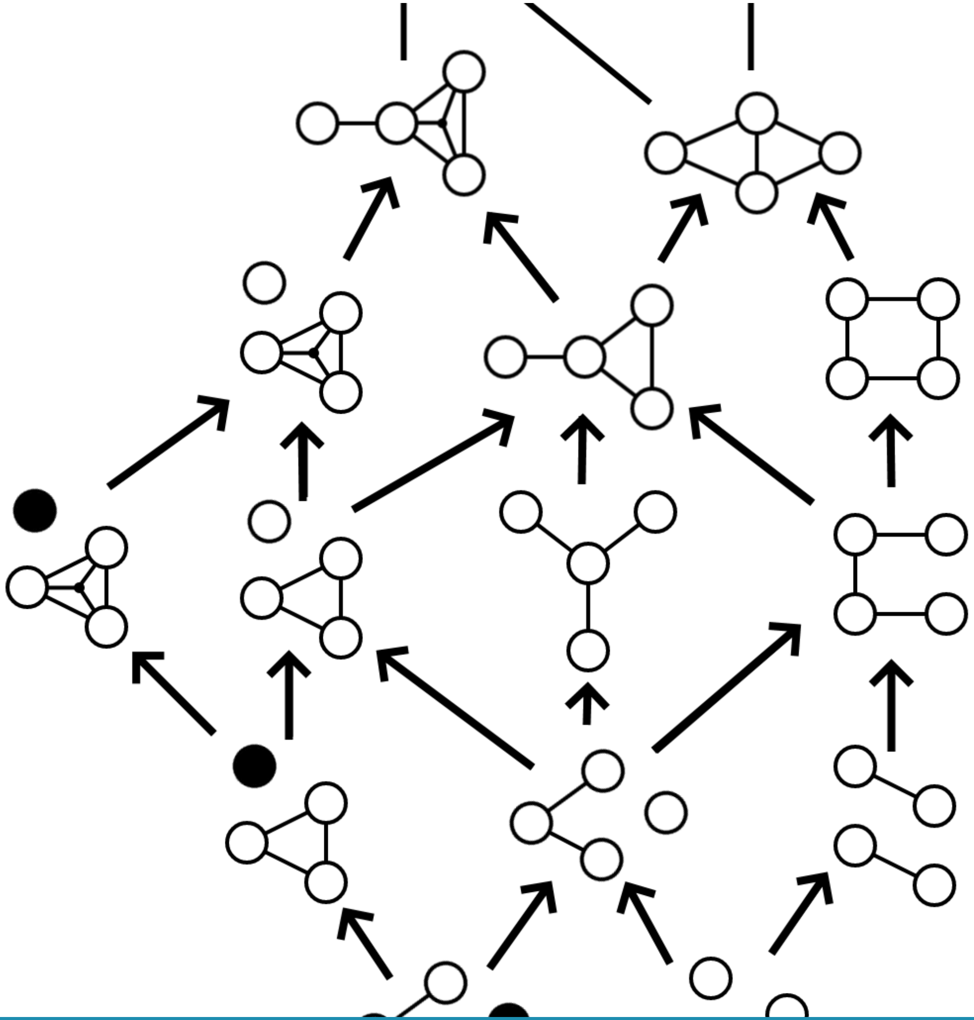
Pad naar $D(9)$

- Efficiënte representatie en operaties
  - → Canonisatie
- Vind alle inequivalente MBFs in 7 variabelen $\Rightarrow$ $(R(7) = 490\,013\,148)$
- Intervalgroottes berekenen $\forall \alpha \in R_7$ bereken $|[\perp, \alpha]|$
  - → Bereken $D(8)$ om te verifiëren
- P-Coefficiënten methode
  - → $D(8)$ Benchmarks
- FPGA Implementatie voor $D(9)$

KU LEUVEN

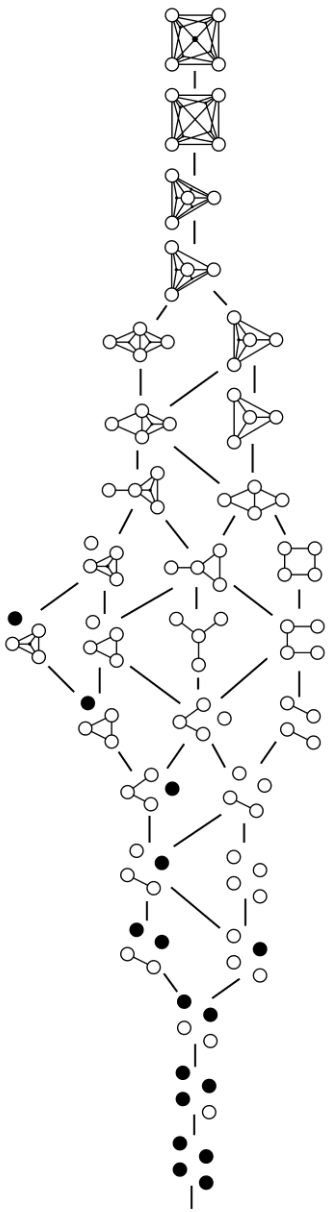
Alle Inequivalente MBFs in 7 variabelen

- Incrementele methode
- Grotere equivalentieklassen door uitbreiding van kleinere
- Sterk afhankelijk van canonisatie performantie



Alle Inequivalente MBFs in 7 variabelen

Methode	Vorige methode: ‘profiles’[1], 2012	Nieuwe Methode
Tijd	8 Maanden	10 minuten
Canonisaties/core/sec	~12’000/sec	~3’000’000/sec
Hardware	meerdere computers	moderne 6-core i7
Canonisaties/klasse	~5000	~22
Memorygebruik	Relatief Klein	Groot
Parallelisatie	Triviaal	Moeilijker
Techniek	Per profile alle equivalentieklassen vinden	Incrementeel kleine equivalentieklassen uitbreiden



1. Tamon Stephen and Timothy Yusun. Counting inequivalent monotone boolean functions, 2012.

KU LEUVEN

Pad naar $D(9)$

- Efficiënte representatie en operaties

  → Canonisatie
- Vind alle inequivalente MBFs in 7 variabelen $\Rightarrow$ $(R(7) = 490\,013\,148)$
- Intervalgroottes berekenen $\forall\alpha \in R_7$ bereken $|[\bot, \alpha]|$

  → Bereken $D(8)$ om te verifiëren
- P-Coefficiënten methode

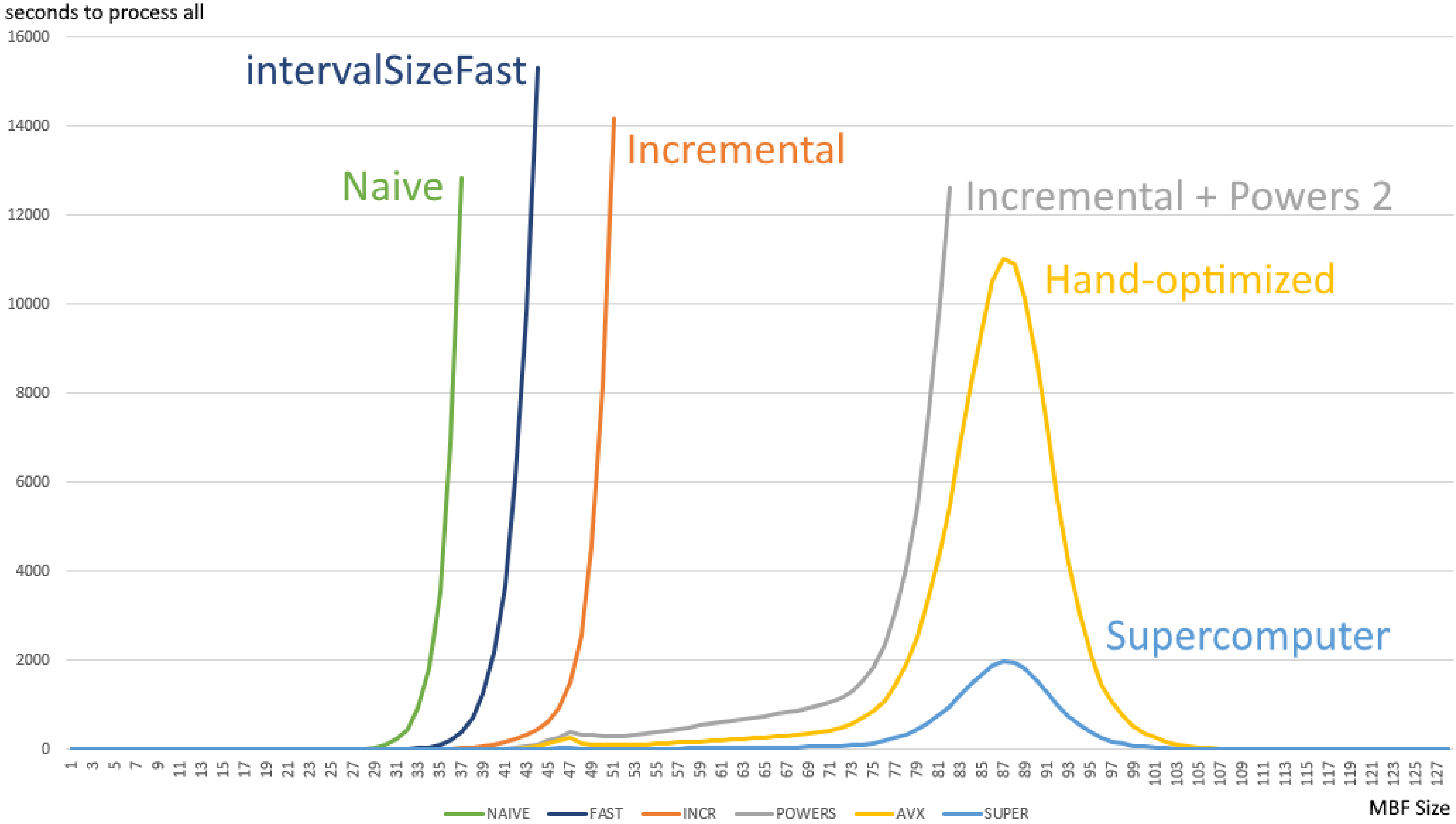
  → $D(8)$ Benchmarks
- FPGA Implementatie voor $D(9)$

KU LEUVEN

Alle intervalgroottes $|[\perp, \alpha]|$ in 7 variabelen

- Incrementeel grotere intervallen berekenen uit kleinere
- Twee belangrijke delen:
  - → Incrementeel bouwen op kleinere intervallen
  - → #Keuzes via machten van 2

Benchmarks



Hand-optimized:
6-core i7: 36 uur

Supercomputer:
36-core Genius server:
6.5 uur

intervalSizeFast: Patrick De Causmaecker and Stefan De Wannemacker. Partitioning in the space of antimonotonic functions, 2011.

Verificatie intervallen door berekening $D(8)$

$$D(n+1) = \sum_{\alpha \in A_n} |[\perp, \alpha]|$$

- Hiermee $D(8)$ berekenen is een nieuw resultaat!

Pad naar $D(9)$

- Efficiënte representatie en operaties
  → Canonisatie
- Vind alle inequivalente MBFs in 7 variabelen $\Rightarrow$ $(R(7) = 490\,013\,148)$
- Intervalgroottes berekenen $\forall \alpha \in R_7$ bereken $|[\bot, \alpha]|$
  → Bereken $D(8)$ om te verifiëren
- P-Coefficiënten methode
  → $D(8)$ Benchmarks
- FPGA Implementatie voor $D(9)$

KU LEUVEN

P-Coëfficiënten methode

$$D(n+k) = \sum_{\substack{\alpha,\beta \in A_n \\ \alpha \le \beta}} |[\bot, \alpha]| P_{n,k,\alpha,\beta} |[\beta, \top]|$$

$$P_{n,2,\alpha,\beta} = 2^{C_{n,\beta-\alpha}}$$

- Zeer krachtig!
- Bereken $D(n+2)$ uit intervallen van n
- $C_{n,\alpha}$: #geconnecteerde componenten in de hypergrafe α
- Zware canonisaties vereist voor opzoeken interval sizes.

Patrick De Causmaecker and Stefan De Wannemacker. On the number of antichains of sets in a finite universe, 2014.

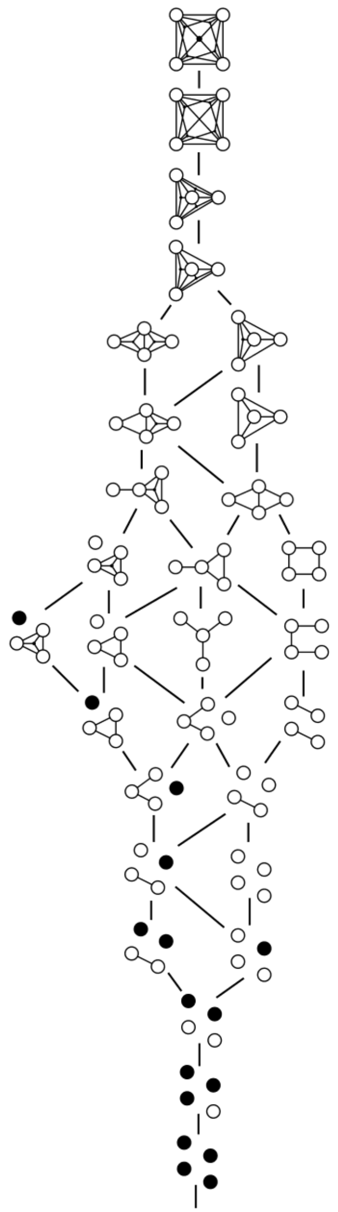
Herwerking van de P-Coëff formule

$$D(n+2) = \sum_{\alpha \in R_n} |[\perp, \alpha]| D_\alpha \sum_{\substack{\beta \in R_n \\ \exists \delta \simeq \beta : \alpha \leq \delta}} |[\beta, \top]| \frac{D_\beta}{n!} \sum_{\substack{\gamma \in Permut_\beta \\ \alpha \leq \gamma}} P_{n,2,\alpha,\gamma}$$

- Vermijd canonisaties door permutaties van gekende equivalentieklassen
- Voor $D(9)$: $\sim 4.59 * 10^{16}$ (α, β) paren
- Gebruik van datastructuur

Efficiente β iteratie datastructuur

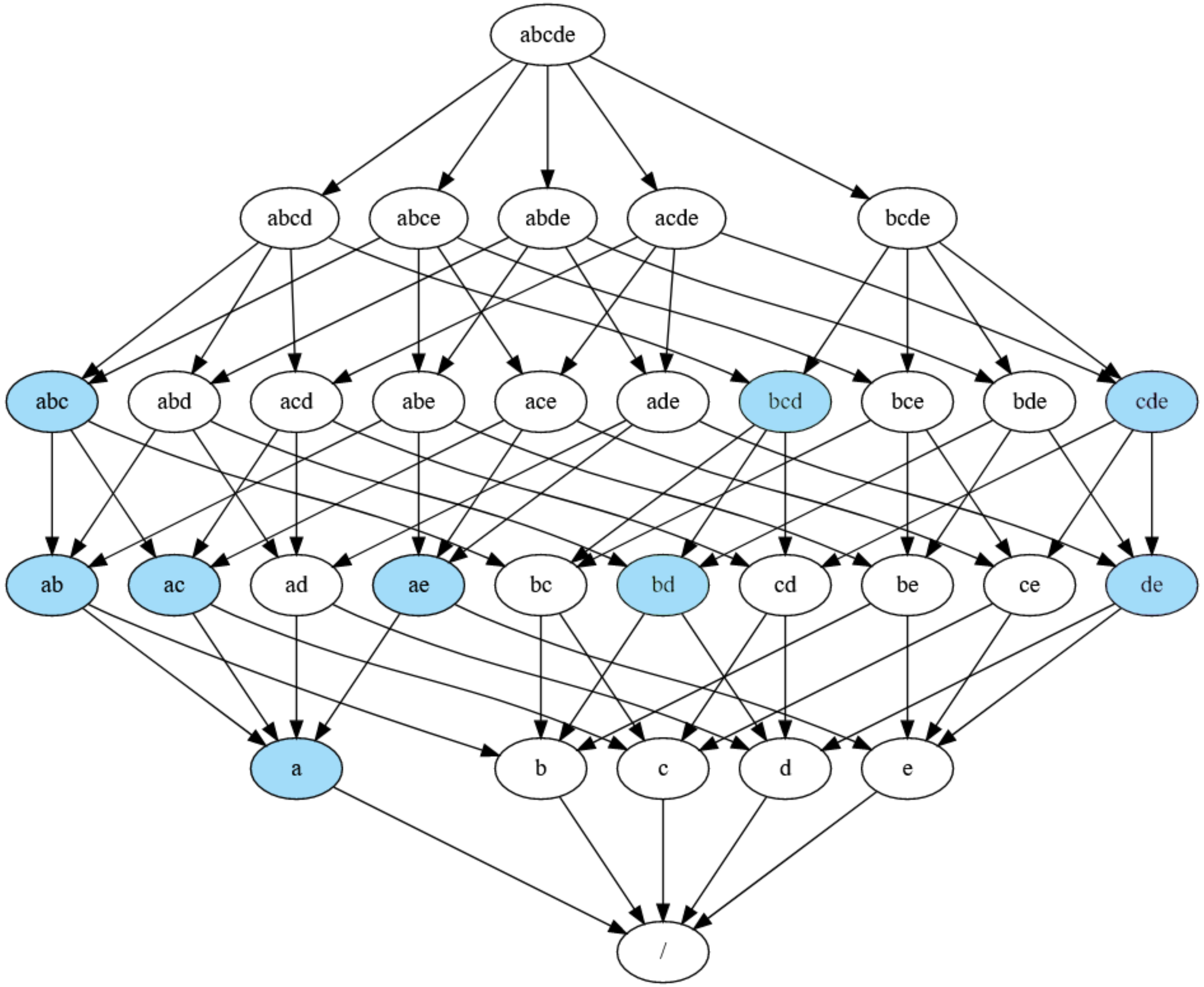
$$\sum_{\substack{\beta \in R_n \\ \exists \delta \simeq \beta : \alpha \leq \delta}}$$



- Link equivalentieklassen aan elkaar
- Pointers volgen ipv canonisaties
- Compacte representatie
- Groot! ~50GB voor 7 variabelen

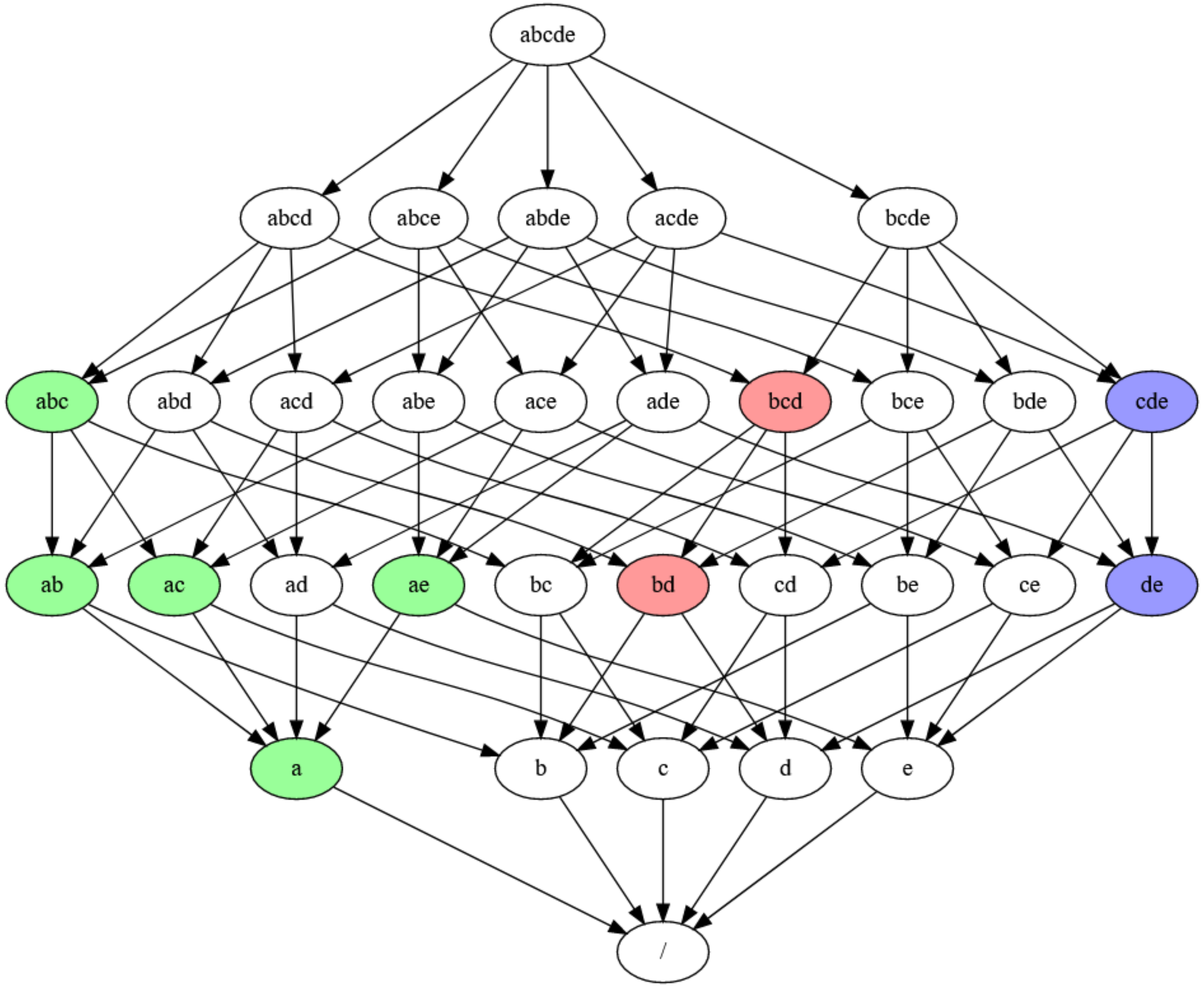
Berekenen van de P-Coëfficiënten

- Tellen van geconnecteerde componenten
- Theoretisch zeer eenvoudig
- Moeilijk te optimaliseren



Berekenen van de P-Coëfficiënten

- Tellen van geconnecteerde componenten
- Theoretisch zeer eenvoudig
- Moeilijk te optimaliseren



Geconnecteerde Componenten Methodes

- Group Merging
  - → Origineel de methode uit de codebase van P. De Causmaecker et al
  - → Maakt weinig gebruik van efficiente operaties
- FloodFill
  - → Gebruik van efficiënte operaties
    - → monotonizeUp & monotonizeDown
  - → Verbeteringen
    - → Singleton Elimination
    - → Leaf Elimination

Patrick De Causmaecker and Stefan De Wannemacker. On the number of antichains of sets in a finite universe, 2014.

$D(8)$ benchmark

Methode	D. Wiedemann	A. Zijlstra	P. De Causmaecker et al	Mijn Implementatie
Tijdsduur	200 uur	30 min	8 uur	6 min
Jaar	1991	2013	2014	2021
Computer	Cray-2 supercomputer	144-core Millipede Cluster	4-core i5	6-core i7
Methode	Wiedemann’s methode	Wiedemann’s methode	P-Coëfficiënten met Antiketens	P-Coëfficiënten met MBFs

Doug Wiedemann. A computation of the eighth dedekind number. Springer Link 10.1007 1991
Arjen Teijo Zijlstra. Calculating the 8th dedekind number. 2013. University of Groningen, Bachelor Thesis
Patrick De Causmaecker and Stefan De Wannemacker. On the number of antichains of sets in a finite universe, 2014.

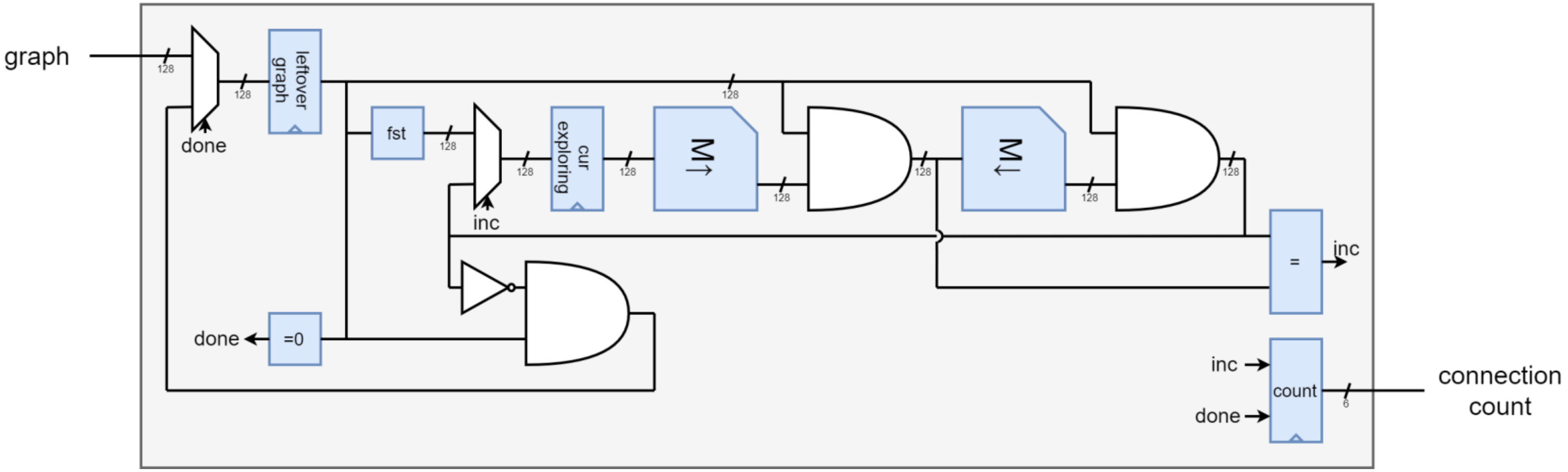
Pad naar $D(9)$

- Efficiënte representatie en operaties
  → Canonisatie
- Vind alle inequivalente MBFs in 7 variabelen ⇒ $(R(7) = 490\ 013\ 148)$
- Intervalgroottes berekenen $\forall \alpha \in R_7$ bereken $|[\bot, \alpha]|$
  → Bereken $D(8)$ om te verifiëren
- P-Coefficiënten methode
  → $D(8)$ Benchmarks
- FPGA Implementatie voor $D(9)$

$D(9)$: P-Coëfficiënten op FPGA

- Floodfill berust op booleaanse operaties en shifts, en is zeer branchy
- Slecht gebruik van CPU
- Niet vertaalbaar naar GPU
- Hardware implementatie!

Count Connected Core

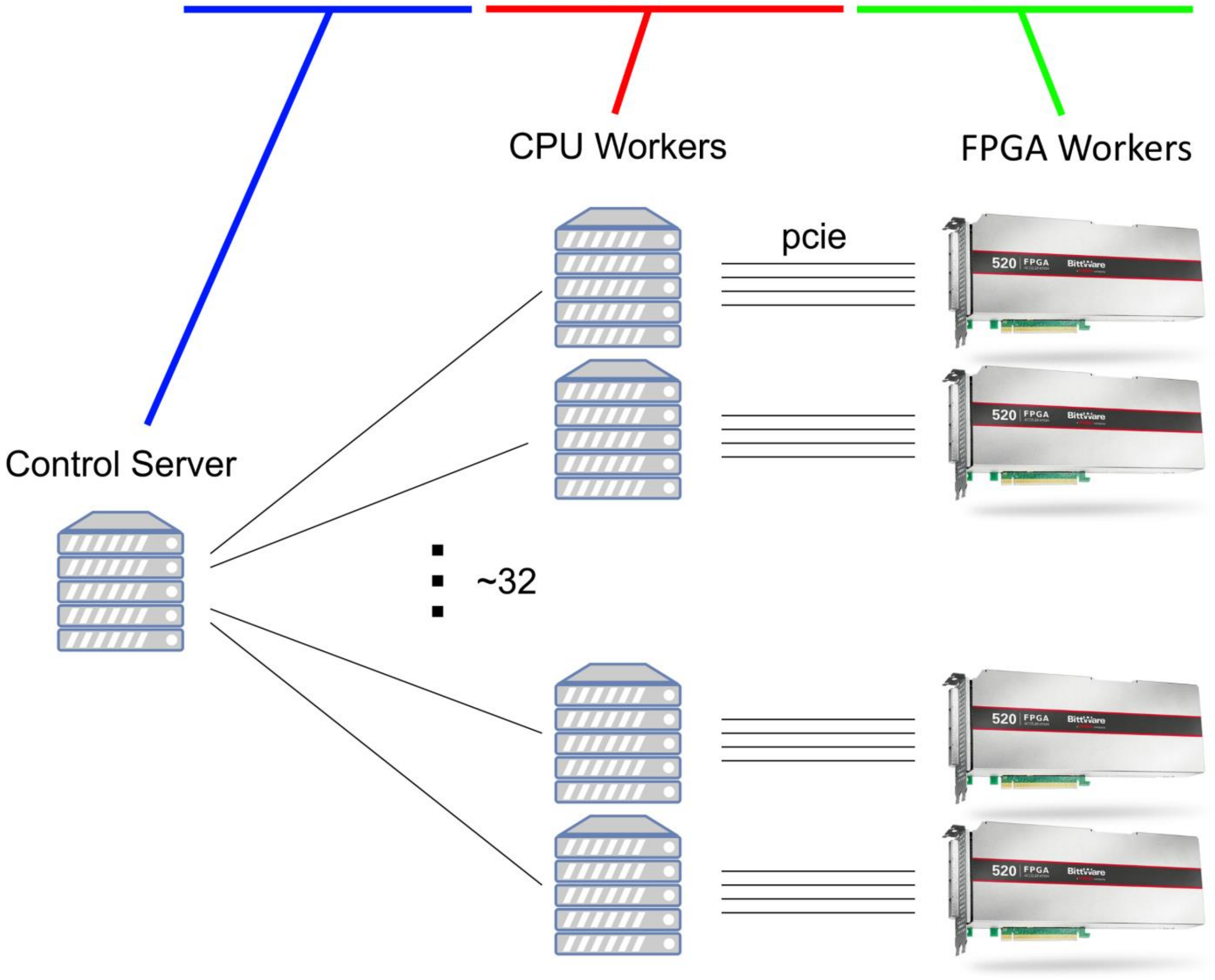


KU LEUVEN

Het Plan voor $D(9)$

$$D(n+2) = \sum_{\alpha\in R_n} |[\bot,\alpha]| D_\alpha \sum_{\substack{\beta\in R_n \\ \exists\delta\simeq\beta:\alpha\leq\delta}} |[\beta,\top]| \frac{D_\beta}{n!} \sum_{\substack{\gamma\in Permut_\beta \\ \alpha\leq\gamma}} P_{n,2,\alpha,\gamma}$$

- ~500x speedup vs CPU
- 32 moderne FPGA kaarten
- Ideaal $D(9)$ 2x op 2,5 maanden



Bemachtigen van de hardware

- 32 dure FPGA kaarten ~13’000$/stuk
  - → Gesprekken met verkopers en VSC
  - → Zoeken naar andere geïnteresseerde onderzoeksgroepen

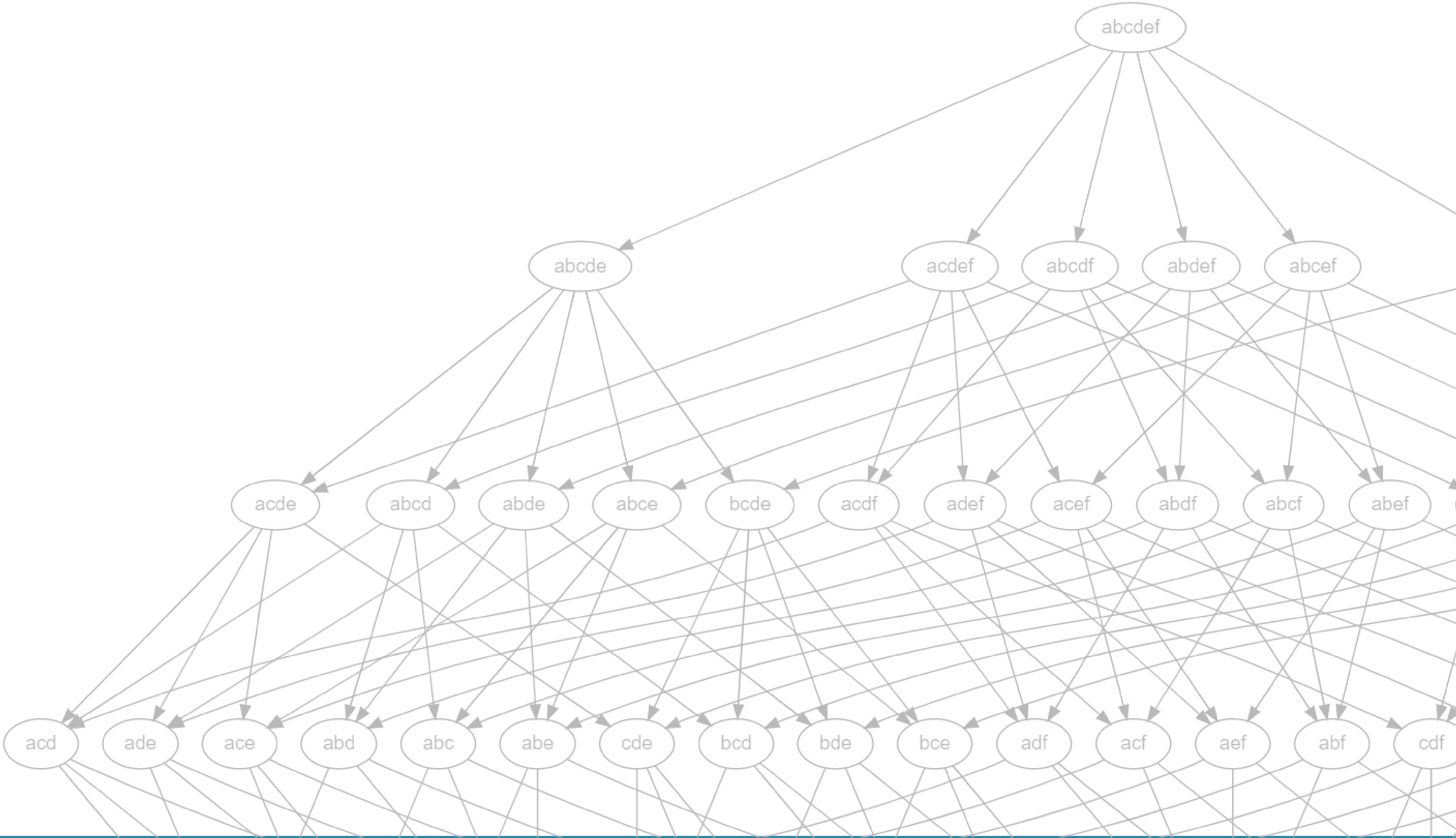
Conclusie

Publicatie-waardige ontwikkelingen:

- Veel sneller algoritme voor $R(7)$

  → Geeft hoop voor $R(8)$
- 1e keer alle intervallen $|[\bot, \alpha]|$ in 7 variabelen berekend
- 1e keer $D(8)$ berekend zonder 2-sprongformule
- 1e haalbare pad voor $D(9)$

Vragen


abcdef
abcde
acdef
abcdf
abdef
abcef
acde
abcd
abde
abce
bcde
acdf
adef
acef
abdf
abcf
abef
acd
ade
ace
abd
abc
abe
cde
bcd
bde
bce
adf
acf
aef
abf
cdf

P-Coëff intuïtie

$$\alpha = \emptyset \times \chi_{\overline{xy}} \;\vee\; \{x\} \times \chi_{x\overline{y}} \;\vee\; \{y\} \times \chi_{\overline{x}y} \;\vee\; \{x, y\} \times \chi_{xy}$$

KU LEUVEN