

# Using Markush search methods for fast calculation of properties in combinatorial libraries

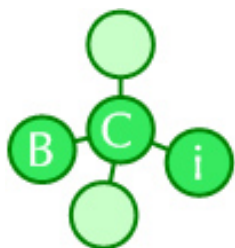
Geoff Downs

Barnard Chemical Information Ltd\*

The Iron Shed, Harewood House Estate, Harewood, Leeds, U.K.

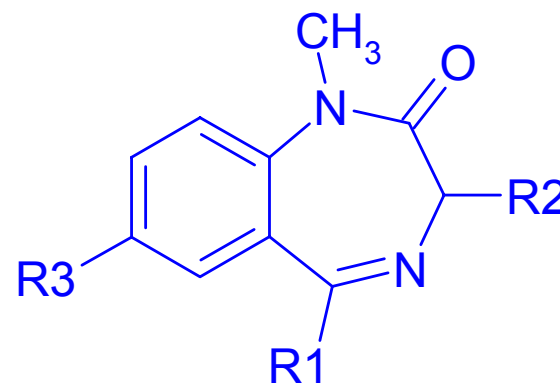
<http://www.bci.gb.com>

\* a subsidiary of Digital Chemistry Ltd  
<http://www.digitalchemistry.co.uk>



# Markush structures

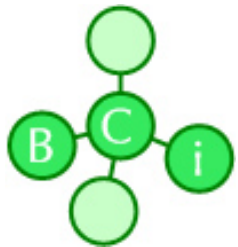
- compact representations of large sets of specific molecules with common structural features
  - ◆ 1 core + 100 R1 + 100 R2 + 100 R3 = 301
  - ◆ 1 core × 100 R1 × 100 R2 × 100 R3 = 1,000,000
- can enumerate specific molecules when needed



R<sub>1</sub> = phenyl / cyclohexyl / ...

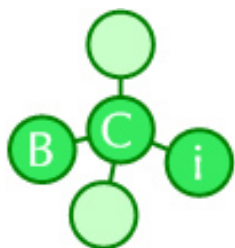
R<sub>2</sub> = H / methyl / ...

R<sub>3</sub> = H / Cl / NO<sub>2</sub> / ...



# Additive property generation

- enumerate specifics; on each specific:
  - ◆ perform fragment search (algorithmic/dictionary-based), or
  - ◆ perform general substructure search
- analyse Markush:
  - ◆ perform fragment search (algorithmic/dictionary-based), or
  - ◆ perform general substructure search
- additive values associated with each definition, fragment or substructure



# Phases of search

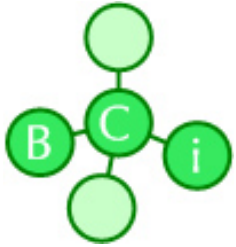
- Searching a Markush allows rapid calculation of many additive features, e.g. fingerprints, properties and counts

## *Sigma Phase ( $\Sigma$ )*

- ◆ Markush is searched for each feature in turn
  - contribution for that feature is added to partial fingerprint, property value etc. for relevant building block(s)

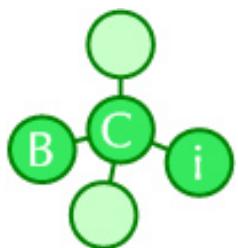
## *Pi Phase ( $\Pi$ )*

- ◆ library enumeration ORs partial fingerprint, sums partial property values etc. for each specific

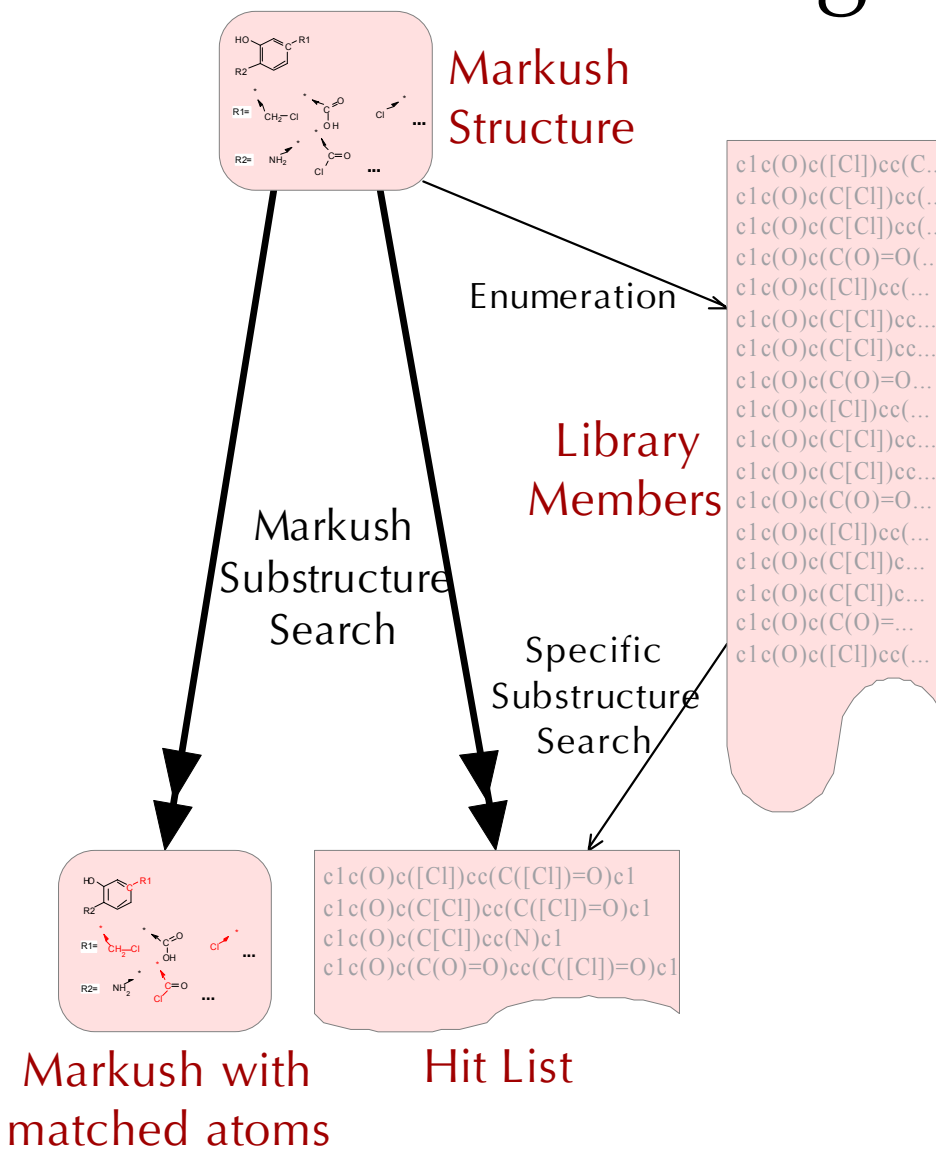


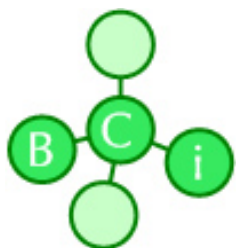
# Fragment-based approach

- addition of hard-coded values, e.g. MW
- coded rules with associated values, e.g. HBA
- algorithmic fragment generators with dictionary of fragments and associated values, e.g. logP
  - ◆ similar to dictionary-based fingerprint generation
- subject of many previous talks
  - ◆ Markush orders of magnitude faster than enumerated specifics route



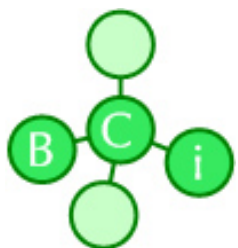
# Substructure searching





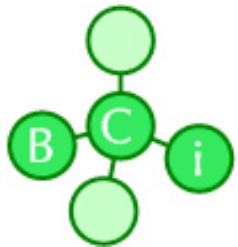
# Substructure searching (Markush)

- direct searching of Markush structure avoids repetition
  - ◆ searches Markush, not specifics
  - ◆ search performed in individual building blocks and at overlaps between them
- substructure search algorithm performs atom-by-atom trace within Markush structure



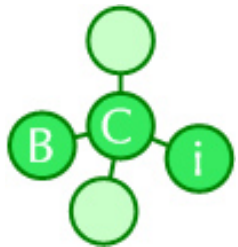
# Query specification

- substructure queries specified as Daylight SMARTS string
  - ◆ can be converted from MDL Molfiles using BCI's MOLSMART program
  - ◆ all Daylight SMARTS features except stereochemistry now implemented, including "recursive" SMARTS
- associate value with each query SMARTS, search Markush and then enumerate property for each specific



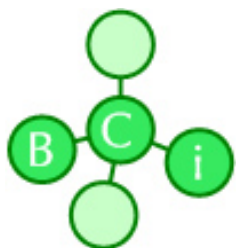
# Comparative timing studies

- study compares time taken to cumulate additive properties using fragment-based (Fragment) and substructure search (SS) on Markush and enumerated specifics
- five properties calculated
  - ◆ HBA, HBD, RB, MR and logP (and all 5 together)
- two series of test datasets used
  - ◆ # Rgroups fixed, # alternatives increases
  - ◆ # Rgroups increases, # alternatives fixed



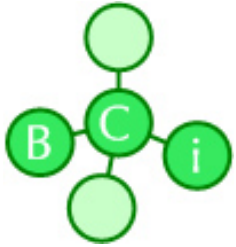
## Test properties (Fragment)

- hard-coded definitions
  - ◆ HBA, HBD & RB
- fragment dictionary definitions
  - ◆ SlogP & MR
  - ◆ 72 atom types (from Wildman & Crippen, JCIICS 39(5), 868-873, 1999)
    - 894 fragments (augmented atom and atom sequence)
    - 6 coded rules



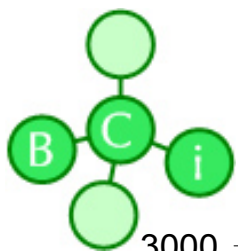
## Test properties (SS)

- SMARTS patterns from ref. 22 in paper by J.S. Delaney, JCICS 44(3), 1000-1005, 2004
  - ◆ HBA – 2 patterns (simple)
  - ◆ HBD – 5 patterns (simple)
  - ◆ RB – 5 patterns (complex)
- SMARTS patterns from Table 1 in paper by Wildman & Crippen, JCICS 39(5), 868-873, 1999
  - ◆ SlogP – 170 patterns (69 atom types)
  - ◆ MR – 131 patterns (65 atom types)
    - 5 patterns don't process

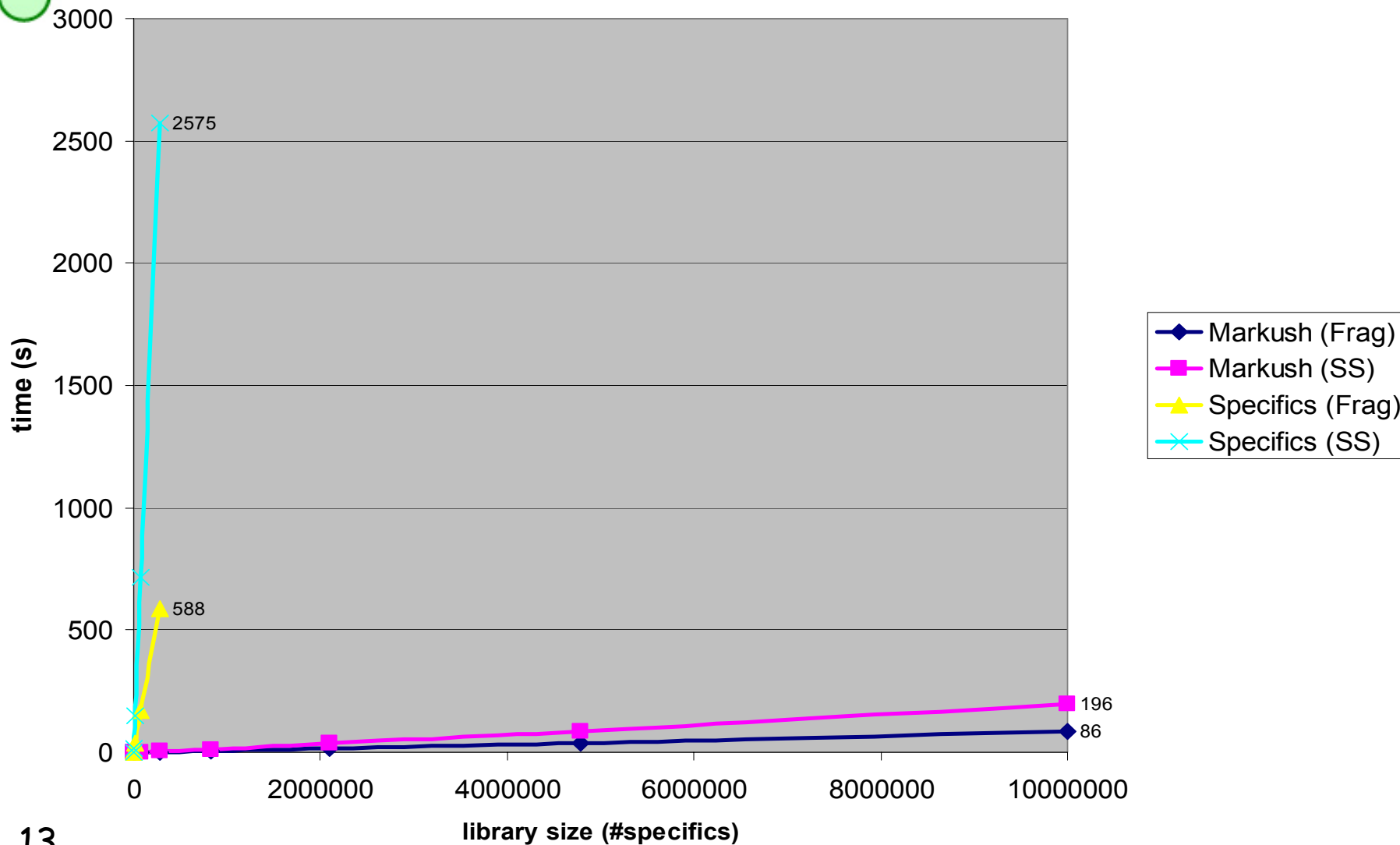


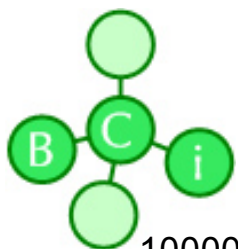
## "# alternatives" test libraries

- 9 subsets of "hugelib"
  - ♦ 7 Rgroups:  $23 \times 22 \times 23 \times 22 \times 16 \times 23 \times 23 = 2,167,088,704$
  - ♦ hl2 = first 2 alternatives =  $2^7 = 128$   
hl3 = first 3 alternatives =  $3^7 = 2,187$   
...  
hl6 = first 6 alternatives =  $6^7 = 279,936$   
end of enumerated specifics tests  
...  
hl10 = first 10 alternatives =  $10^7 = 10,000,000$   
end of Markush tests
- 2.8MHz Pentium 4 (512Mb); debug code!

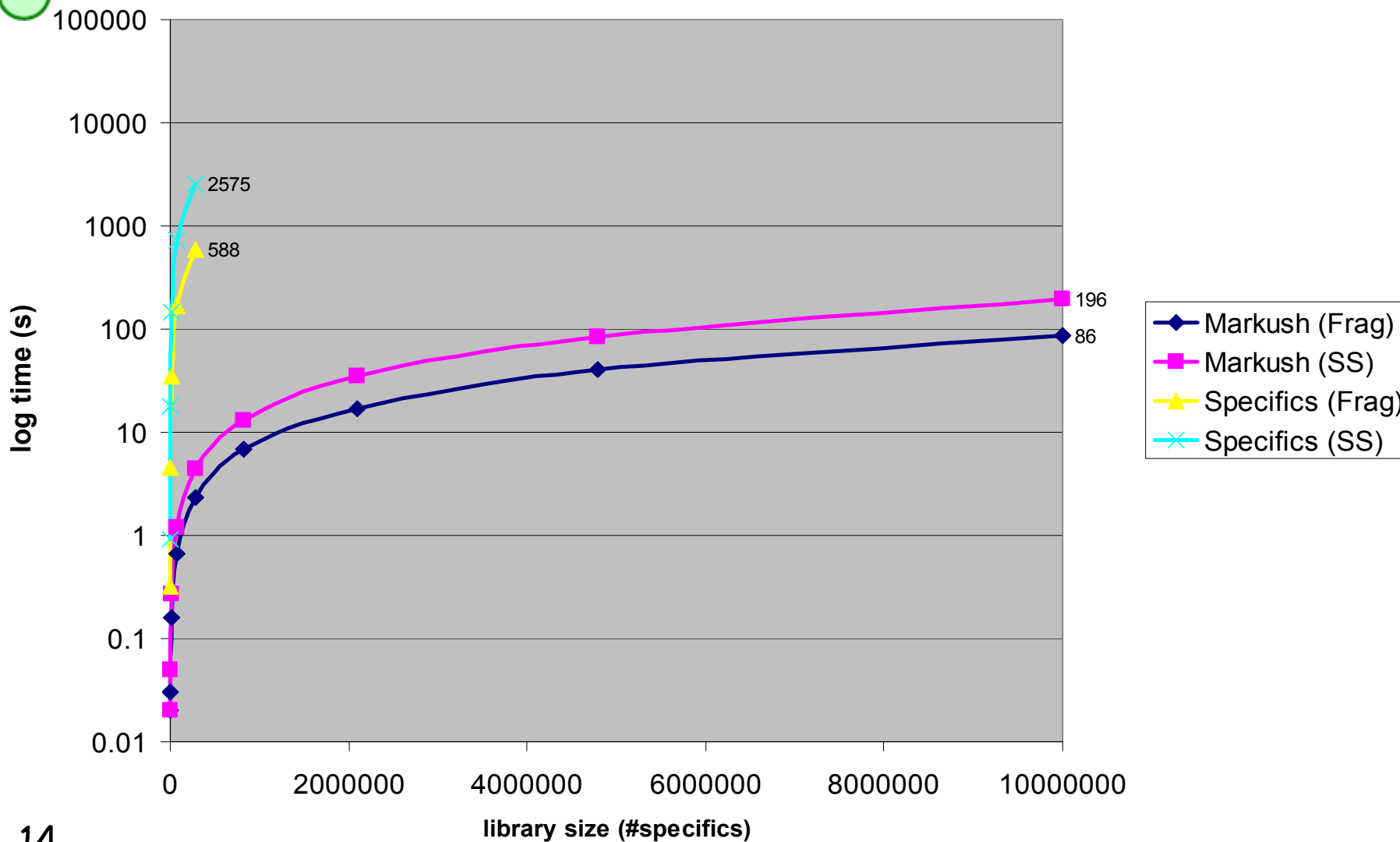


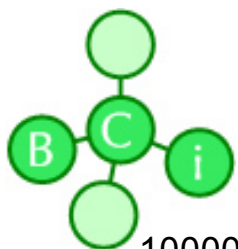
# RB timings (& HBA/HBD)



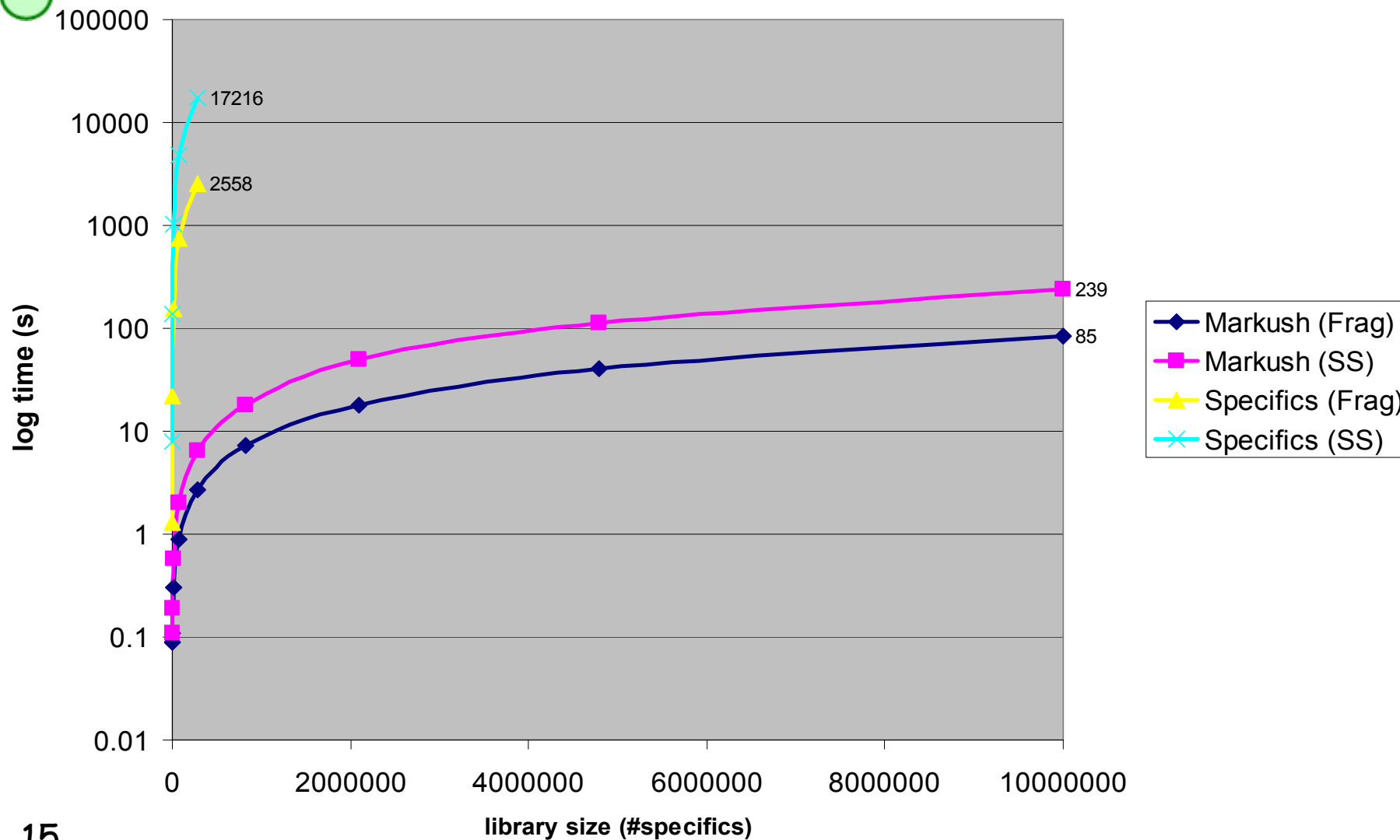


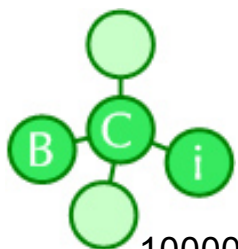
# RB timings (& HBA/HBD)



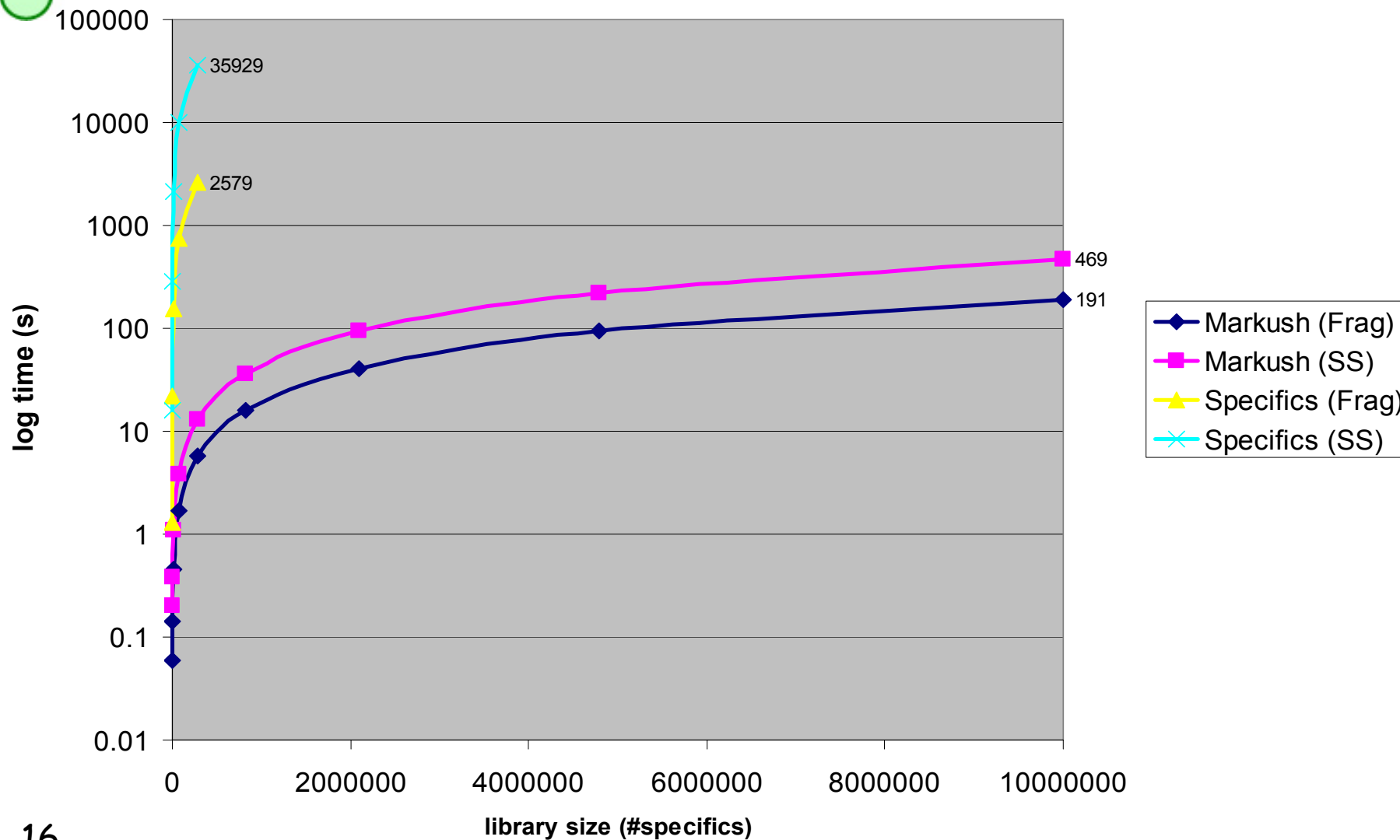


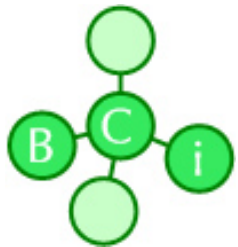
# SlogP timings (& MR)





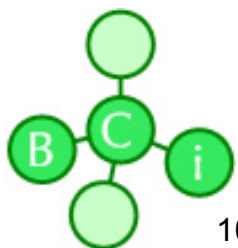
# Timings for all 5 properties



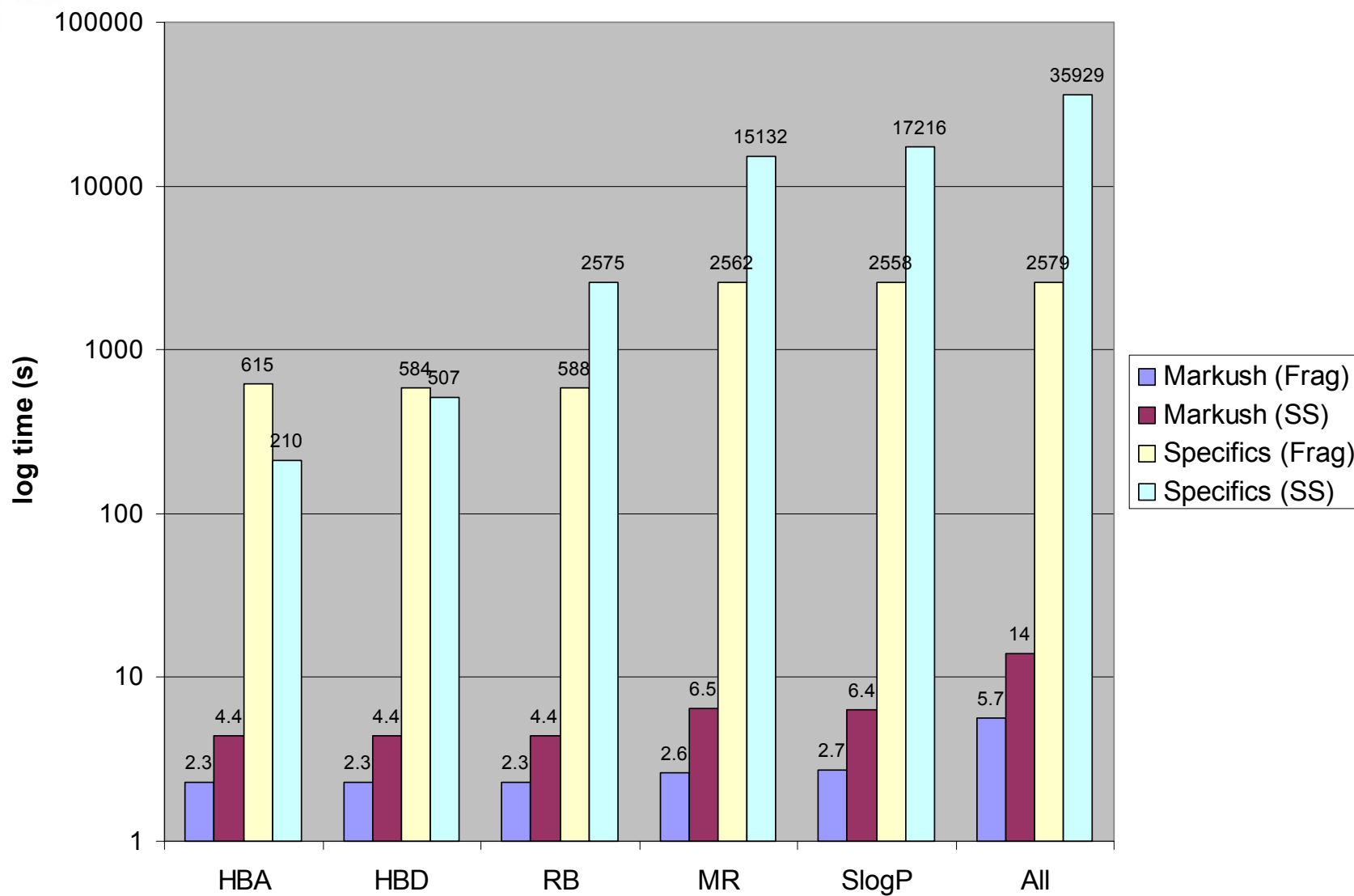


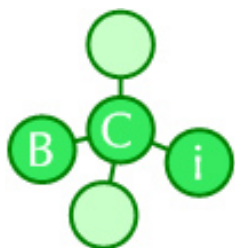
# Summary of library timings (# alternatives)

- linear increase in property generation times with increasing library size
- Markush Fragment fastest; specifics SS slowest
  - ◆ e.g. for hl6 (~280,000 specifics), all properties:
    - Markush Fragment : < 6s
    - Markush SS : 14s
    - specifics Fragment : 43 min
    - specifics SS : 10 hours



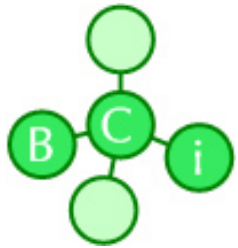
# hl6 timings





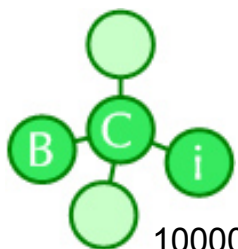
## Summary of property timings (# alternatives)

- HBA/HBD/RB about the same
  - ◆ Specifics SS sometimes faster than Fragment
- SlogP/MR about the same; slower than HBA etc. but not in proportion to number of fragments for Markush
- overall, Markush Fragment fastest; specifics SS slowest
- calculating all 5 properties:
  - ◆ Markush Fragment & SS = SlogP + HBA times
  - ◆ specifics Fragment = SlogP times
  - ◆ specifics SS = HBA+HBD+RB+SlogP+MR times

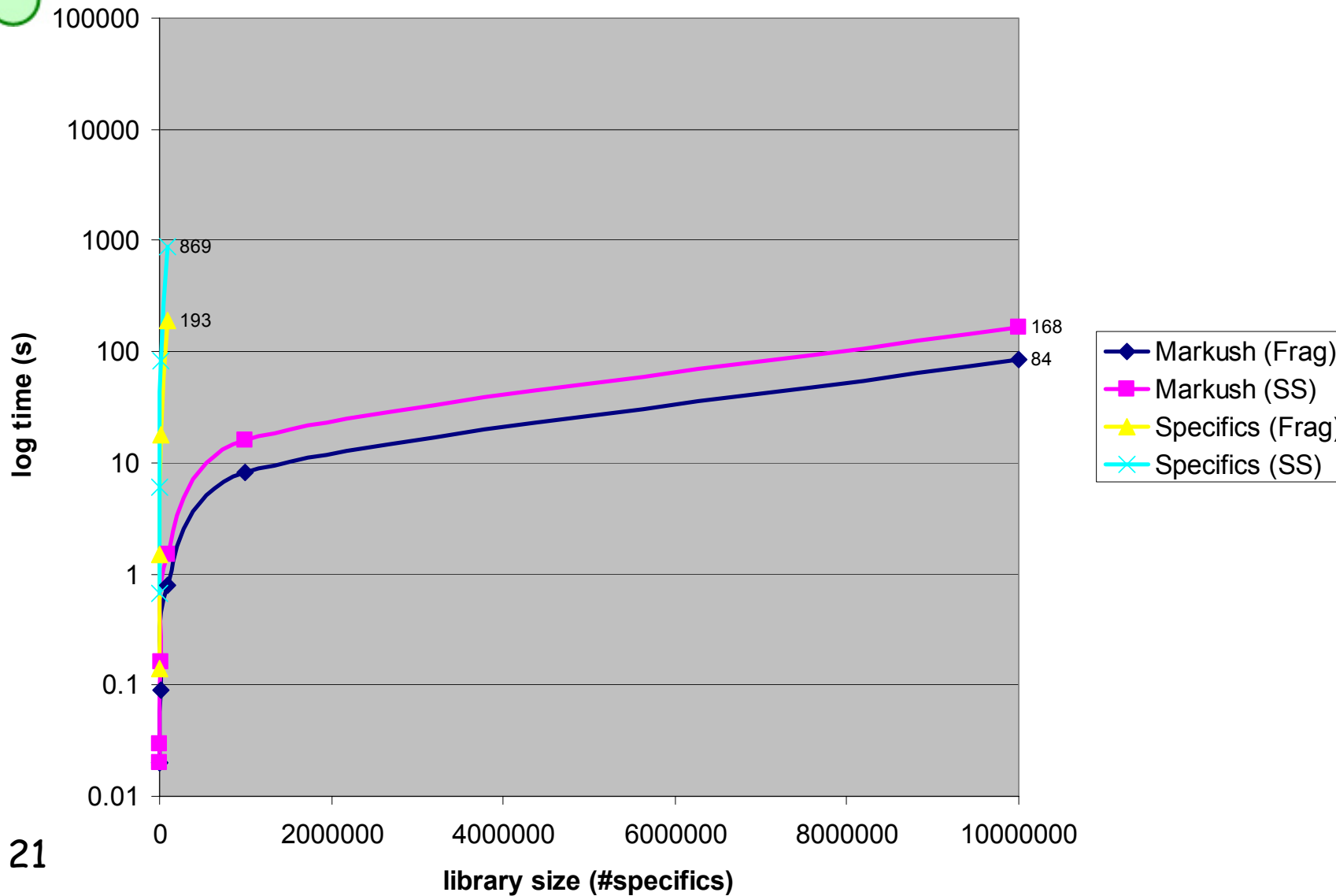


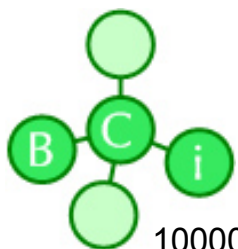
## "# Rgroups" test libraries

- 5 subsets of hl10 (& hl10 itself)
  - ◆ 10 alternatives for each Rgroup
  - ◆ hl10\_2 = 2 Rgroups (100 specifics)
  - ...
  - hl10\_5 = 5 Rgroups (100,000 specifics)
  - end of enumerated specifics tests
  - ...
  - hl10\_7 = hl10 (all 7 Rgroups; 10,000,000 specifics)
  - end of Markush tests

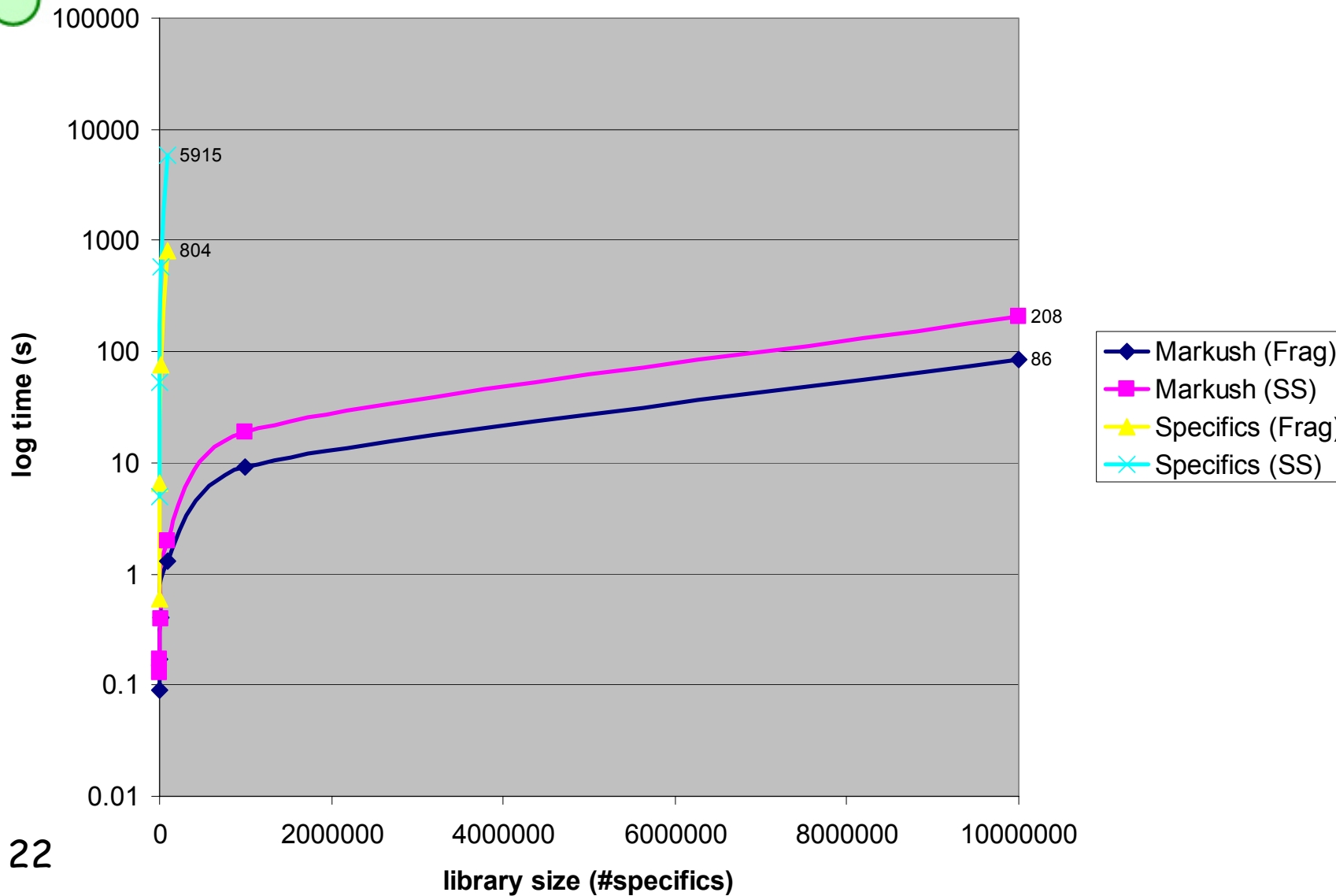


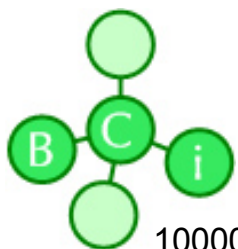
# RB timings (& HBA/HBD)



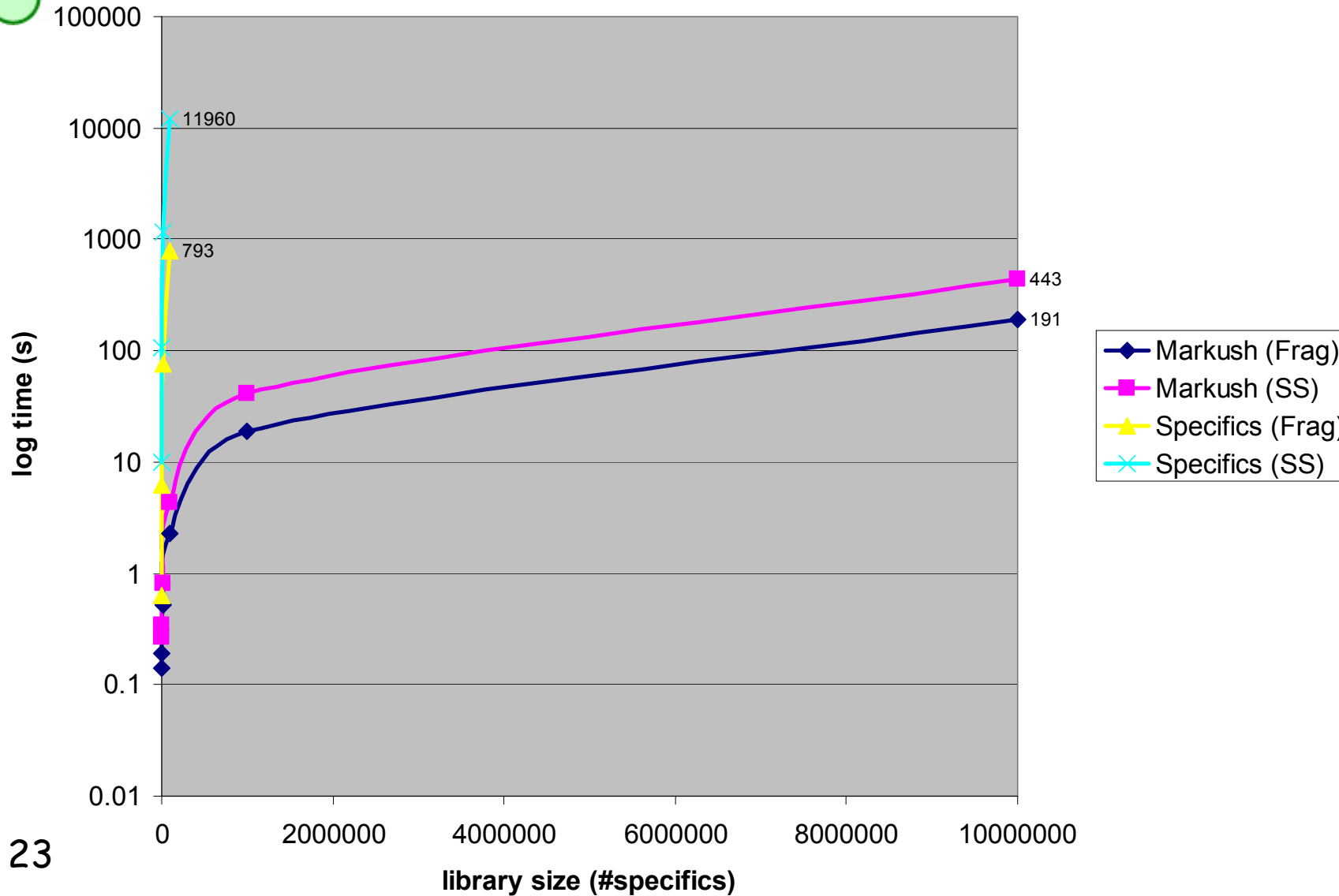


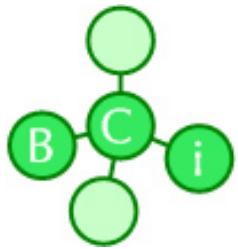
# SlogP timings (& MR)





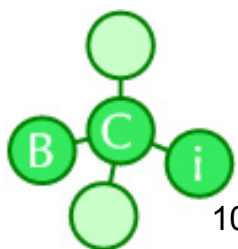
# Timings for all 5 properties



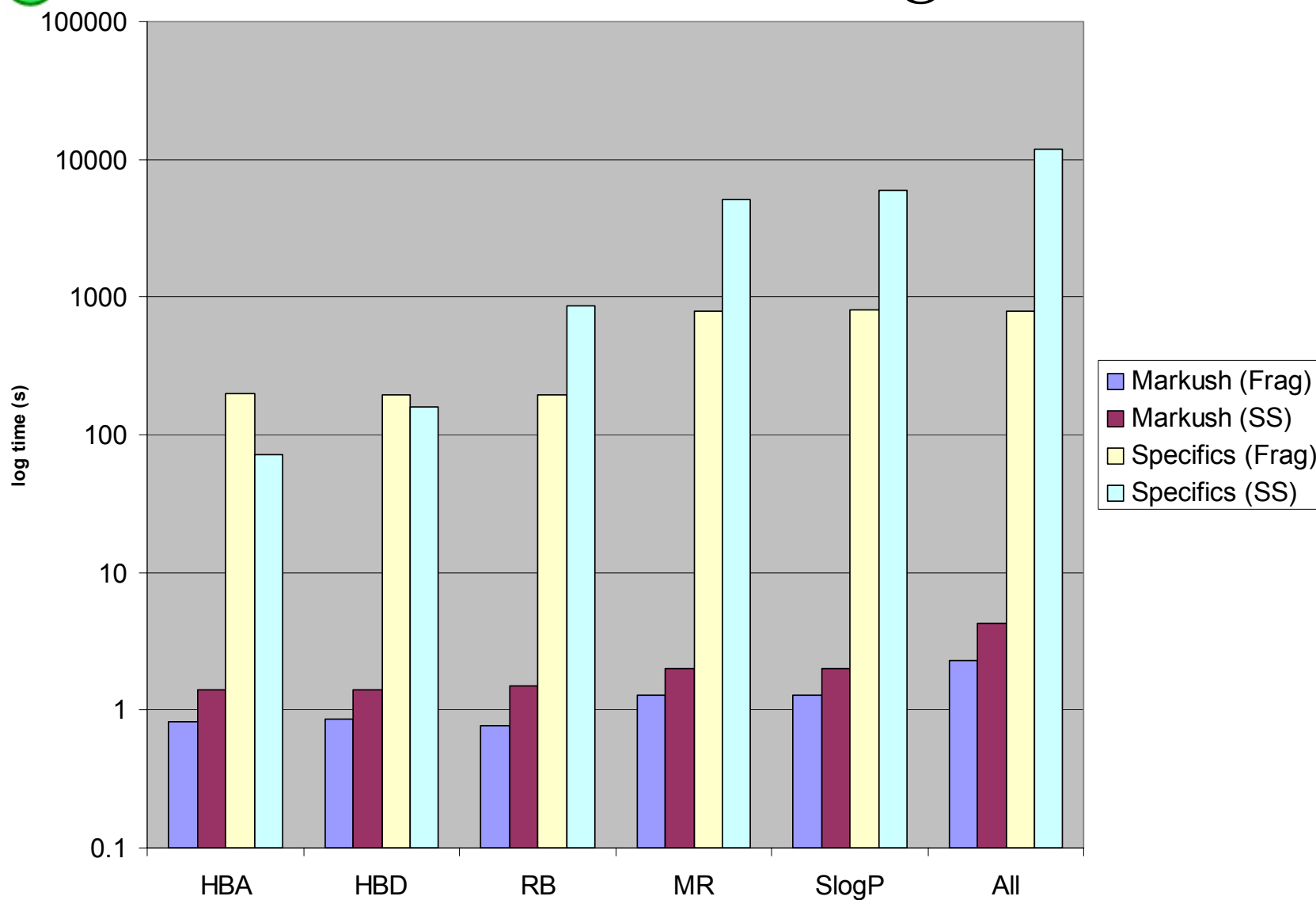


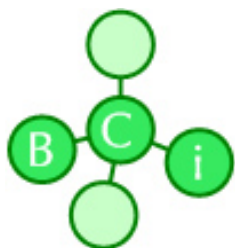
# Summary of library timings (# Rgroups)

- Linear increase in property generation times with increasing library size after about 10K
  - ◆ times more erratic for < 10K
- Markush Fragment fastest; specifics SS slowest
  - ◆ e.g. for hl10\_5 (100,000 specifics), all properties:
    - Markush Fragment : ~ 2.5s
    - Markush SS : ~4.5s
    - specifics Fragment : ~13 min
    - specifics SS : ~3 hours 20 min



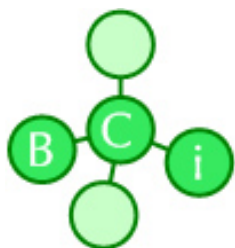
# hl10\_5 timings





# Summary of property timings (# Rgroups)

- HBA/HBD/RB about the same
  - ◆ SS can be faster than Fragment for small libraries
- SlogP/MR about the same; slower than HBA etc. but not in proportion to number of fragments
- overall, Markush Fragment fastest; specifics SS slowest
- calculating all 5 properties:
  - ◆ Markush Fragment & SS = SlogP + HBA times
  - ◆ specifics Fragment = SlogP times
  - ◆ specifics SS = HBA+HBD+RB+SlogP+MR times



## Overall summary

- Markush methods for additive property generation much faster than specifics
  - ◆ fragment-based faster than substructure search
- linear increase in processing times as dataset size increases
- for Markush, no difference between varying #alternatives or #Rgroups
  - ◆ e.g. “all property” plots overlay each other
- for Markush, sub-linear increase in processing time as number of fragments/substructures increases