EFFICIENT MATCHING OF MULTIPLE CHEMICAL SUBGRAPHS

ROGER SAYLE

NEXTMOVE SOFTWARE LTD
CAMBRIDGE UK

9th ICCS, Noordwijkerhout, The Netherlands, 9th June 2011

OVERVIEW

- Chemical Pattern Matching
- Efficient Single Pattern Matching
- Multiple Pattern Matching
- Toolkit Code Generation
- Performance Figures
- Conclusions

PREVIOUS WORK

- Efficient Protein and Nucleic Acid Perception from Simple Atomic Connectivity

  www.daylight.com/meetings/mug96/sayle/sayle.html

  Describes algorithms for perceiving protein sequence and PDB atom names from SMILES, mol or XYZ file of a protein.

- 1st Class SMARTS patterns

  www.daylight.com/meetings/emug97/Sayle/

  Describes SMARTS syntax and SMARTS algebra, a set of semantics preserving transformations that can be used to optimize SMARTS patterns.

CHEMICAL PATTERN MATCHING

- The identification of a specific subgraph within a graph, also known as *subgraph isomorphism*
- Typically to identify a functional group or substructure in a molecule connection table.
- Query patterns are typically specified as SMARTS, MDL query files, CDX or Marvin files.
- Matching is performed using
  - Ullman’s isomorphism algorithm [1970]
  - McGregor’s backtracking search [1981]

CHEMICAL DATABASE SEARCHING

- Although a backtracking atom-by-atom match is very efficient for matching a single pattern against a single molecule, well known optimizations exist for scanning a large database of target molecules.
  - Fingerprint screening/inverted indices
  - Character frequency (histogram) screening
  - "Triage" substructure identification

  http://www.daylight.com/meetings/emug00/Sayle/substruct.html

TOOLKIT (SMARTS?) PERFORMANCE

- Time taken to find O=[C,N]aa[N,O;!H0] hits in 250,251 SMILES of the NCI August 2000 data.

ToolKit	Times (secs)
OpenEye OEChem v1.7	8.6
ChemAxon JChem v5.5	58.8
RDKit v2011_03_2	131.2
OpenBabel v2.3.0	272.5
PerlMol	2107.9
CDK v1.2.10	DNF

Cheminformatics Applications

- Compound Filtering
- Fingerprint generation
  - Database clustering
- Atom Typing
  - Property prediction

Filtering Radioactive Compounds

- A molecule is radioactive if any of its atoms are radioactive. An atom is radioactive if its is not "stable".
- If an isotope is specified it must be one of the 255 known stable nuclides, otherwise the corresponding element must have at least one stable isotope.
- Elements H to $_{82}\mathrm{Pb}$, with exceptions of $_{43}\mathrm{Tc}$ and $_{61}\mathrm{Pm}$.
- Hence stable is "[0#1,1#1,2#1,0#2,3#2,4#2…]".
- Hence, radioactive is "[!0,!#1;!1,!#1;!2,!#2;…]".

RADIOACTIVE SMARTS

```
[!0,!#1;!1,!#1;!2,!#1;!0,!#2;!3,!#2;!4,!#2;!0,!#3;!6,!#3;!7,!#3;!0,!#4;!9,!#4;!0,!#5;!10,!#5;!11,!#5;!0,!#6;!12,!#6;!13,!#6;!0,!#7;!14,!
#7;!15,!#7;!0,!#8;!16,!#8;!17,!#8;!18,!#8;!0,!#9;!19,!#9;!0,!#10;!20,!#10;!21,!#10;!22,!#10;!0,!#11;!23,!#11;!0,!#12;!24,!#12;!25,!
#12;!26,!#12;!0,!#13;!27,!#13;!0,!#14;!28,!#14;!29,!#14;!30,!#14;!0,!#15;!31,!#15;!0,!#16;!32,!#16;!33,!#16;!34,!#16;!36,!#16;!0,
!#17;!35,!#17;!37,!#17;!0,!#18;!36,!#18;!38,!#18;!40,!#18;!0,!#19;!39,!#19;!41,!#19;!0,!#20;!40,!#20;!42,!#20;!43,!#20;!44,!#20;
!46,!#20;!0,!#21;!47,!#21;!0,!#22;!46,!#22;!47,!#22;!48,!#22;!49,!#22;!50,!#22;!0,!#23;!51,!#23;!0,!#24;!50,!#24;!52,!#24;!53,!#
24;!54,!#24;!0,!#25;!55,!#25;!0,!#26;!54,!#26;!56,!#26;!57,!#26;!58,!#26;!0,!#27;!59,!#27;!0,!#28;!58,!#28;!60,!#28;!61,!#28;!62,
!#28;!64,!#28;!0,!#29;!63,!#29;!65,!#29;!0,!#30;!64,!#30;!66,!#30;!67,!#30;!68,!#30;!70,!#30;!0,!#31;!69,!#31;!71,!#31;!0,!#32;!
70,!#32;!72,!#32;!73,!#32;!74,!#32;!0,!#33;!75,!#33;!0,!#34;!74,!#34;!76,!#34;!77,!#34;!78,!#34;!80,!#34;!0,!#35;!79,!#35;!81,!#
35;!0,!#36;!79,!#36;!80,!#36;!82,!#36;!83,!#36;!84,!#36;!86,!#36;!0,!#37;!85,!#37;!0,!#38;!84,!#38;!86,!#38;!87,!#38;!88,!#38;!0,
!#39;!89,!#39;!0,!#40;!90,!#40;!91,!#40;!92,!#40;!94,!#40;!96,!#40;!0,!#41;!93,!#41;!0,!#42;!92,!#42;!94,!#42;!95,!#42;!96,!#42;
!97,!#42;!98,!#42;!0,!#44;!96,!#44;!98,!#44;!99,!#44;!100,!#44;!101,!#44;!102,!#44;!104,!#44;!0,!#45;!103,!#45;!0,!#46;!102,!#4
6;!104,!#46;!105,!#46;!106,!#46;!108,!#46;!110,!#46;!0,!#47;!107,!#47;!109,!#47;!0,!#48;!106,!#48;!108,!#48;!110,!#48;!111,!#
48;!112,!#48;!114,!#48;!0,!#49;!113,!#49;!0,!#50;!112,!#50;!114,!#50;!115,!#50;!116,!#50;!117,!#50;!118,!#50;!119,!#50;!120,!
#50;!122,!#50;!124,!#50;!0,!#51;!121,!#51;!123,!#51;!0,!#52;!120,!#52;!122,!#52;!123,!#52;!124,!#52;!125,!#52;!126,!#52;!0,!#5
3;!127,!#53;!0,!#54;!124,!#54;!126,!#54;!128,!#54;!129,!#54;!130,!#54;!131,!#54;!132,!#54;!134,!#54;!136,!#54;!0,!#55;!133,!#
55;!0,!#56;!130,!#56;!132,!#56;!134,!#56;!135,!#56;!136,!#56;!137,!#56;!138,!#56;!0,!#57;!139,!#57;!0,!#58;!136,!#58;!138,!#5
8;!140,!#58;!142,!#58;!0,!#59;!141,!#59;!0,!#60;!142,!#60;!143,!#60;!145,!#60;!146,!#60;!148,!#60;!0,!#62;!144,!#62;!149,!#62;
!150,!#62;!152,!#62;!154,!#62;!0,!#63;!153,!#63;!0,!#64;!154,!#64;!155,!#64;!156,!#64;!157,!#64;!158,!#64;!160,!#64;!0,!#65;!1
59,!#65;!0,!#66;!156,!#66;!158,!#66;!160,!#66;!161,!#66;!162,!#66;!163,!#66;!164,!#66;!0,!#67;!165,!#67;!0,!#68;!162,!#68;!16
4,!#68;!166,!#68;!167,!#68;!168,!#68;!170,!#68;!0,!#69;!169,!#69;!0,!#70;!168,!#70;!170,!#70;!171,!#70;!172,!#70;!173,!#70;!1
74,!#70;!176,!#70;!0,!#71;!175,!#71;!0,!#72;!176,!#72;!177,!#72;!178,!#72;!179,!#72;!180,!#72;!0,!#73;!180,!#73;!181,!#73;!0,!
#74;!182,!#74;!183,!#74;!184,!#74;!186,!#74;!0,!#75;!185,!#75;!0,!#76;!184,!#76;!187,!#76;!188,!#76;!189,!#76;!190,!#76;!192,
!#76;!0,!#77;!191,!#77;!193,!#77;!0,!#78;!192,!#78;!194,!#78;!195,!#78;!196,!#78;!198,!#78;!0,!#79;!197,!#79;!0,!#80;!196,!#80
;!198,!#80;!199,!#80;!200,!#80;!201,!#80;!202,!#80;!204,!#80;!0,!#81;!203,!#81;!205,!#81;!0,!#82;!204,!#82;!206,!#82;!207,!#8
2;!208,!#82]
```

COMPARATIVE PERFORMANCE

	OEChem 1.7 (C++)		ChemAxon 5.5 (Java)	
	Total	Match	Total	Match
Time to read file	8.05s		47.57s	
Time to match '[!0]'	8.32s	0.27s	53.74s	6.17s
`radioactive.sma`	102.60s	94.55s	65.05s	17.48s

* Issues with OpenBabel, RDKit and CDK

Interpreters vs Compilers

- Most SMARTS matchers are implemented as “interpreters”, that parse the SMARTS string at run-time, build an internal parse tree, and then repeatedly traverse this at match-time.
- This is analogous to Perl and Python.
- For static patterns, the SMARTS may be compiled (parsed and optimized) ahead of time for faster execution at match-time.

Generated OEChem Source Code

```
#include <oechem.h>

bool isRadioactive(const OEChem::OEMolBase &mol) {
  OESystem::OEIter<OEChem::OEAtomBase> atom;
  for (atom = mol.GetAtoms(); atom; ++atom) {
    const OEChem::OAtomBase *aptr = atom;
    switch (aptr->GetAtomicNum()) {
    case 1:
      switch (aptr->GetIsotope()) {
      case 0:
      case 1:
      case 2:
        break;
      default:
        return true;
```

Generated JChem Source Code

```
static boolean isRadioactive(chemaxon.struc.Molecule mol) {
  int count = mol.getAtomCount();
  for (int i=0; i<count; i++) {
    chemaxon.struc.MolAtom atom = mol.getAtom(i);
    switch (atom.getAtno()) {
    case 1:
      switch (atom.getMassno()) {
      case 0:
      case 1:
      case 2:
        break;
      default:
        return true;
      }
      break;
```

Compiled Matching Performance

	OEChem 1.7 (C++)		ChemAxon 5.5 (Java)	
	Total	Match	Total	Match
Time to read file	8.05s		47.57s	
Time to match '[!0]'	8.32s	0.27s	53.74s	6.17s
`radioactive.sma`	102.60s	94.55s	65.05s	17.48s
isRadioactive code	8.21s	0.16s	47.60s	0.03s
Speed-up	~12.5x	~580x	~1.38x	~700x

PATSY "BACKEND" TARGETS

- OEChem (C++/Python/Java)
- OpenBabel (C++)
- RDKit (C++/Python)
- CDK (Java)
- Cinfony/Pybel (Python)
- PerlMol (Perl)
- Pipeline Pilot (PilotScript)
- Isentris (Cheshire)

ISOMORPHISM COUNTING

- Match counts are frequently used in filtering apps.
- A benchmark of match iteration performance is to count the isomorphisms of ferrocene (to itself).
- SMILES: C12C3[Fe]1456789(C2C4C5C3)C1C6C7C8C91
- The correct answer is 200.
- OpenEye’s OEChem v1.7 takes 3.461s
- A (Patsy) compiled C++ matcher using OEChem v1.7 takes only 0.008s.

MATCH GENERATORS

- Almost all cheminformatics toolkits provide a mechanism for returning the set of matchings of a given query pattern against a target.
- Most such implementations are “eager”; determining all solutions in advance.
- A more efficient, but technically more challenging solution, is to provide a “lazy” iterator (called a generator in python) reducing both run-time and memory.

Tripos Sybyl Atom Types

```
# Carbon atom types
[#6+]                           C.cat
[c]                             C.ar
[$(C#*),$(C(=*)=*)]             C.sp
C=*                             C.sp2
[#6]                            C.sp3

#Oxygen atom types
[$(OC=O),$(O=CO)]               O.co2
O=*                             O.2
[#8]                            O.3
```

```
# Nitrogen atom types
[nX2]                           N.ar
[#7X4]                          N.4
NC=[O,S]                        N.am
[NX2]=*                         N.2
[$(N=*),$(N*=*),$(Na),n]        N.pl3
[#7]                            N.3

# Hydrogen atom type
[#1]                            H
```

Merck MMFF94 Atom Types

- Halgren’s MMFF94 forcefields use 216 atom types internally [96 external atom types]

[OX1]=[#6]~[#7]	O=CN
[OX1]=[#6]~[#8]	O=CO
[OX1]=[#6]~[#6]	O=CR
[OX1]=[#6]	O=C
[OX1]=[#7]	O=N
[OX1]=S=*	O=S=
[OX1]=S	O=S

IUPAC SUFFIX ATOM TYPES

- OpenEye’s Lexichem internally uses a system of ~1034 atom types for determining the principal suffix during IUPAC name generation.

[CX3](=[OX1])-[OX1-]	carboxylate
[CX3](=[OX1])-[OHX1]	carboxylic acid
[CX3](=[OX1])-[OX2]	carboxylic acid ester
[CX3](=[OX1])-[NX3]	amide
[CX3](=[OX1])-[NX3]-[NX3]	hydrazide
[CX3H]=[OX1]	aldehyde
[CX3H0]=[OX1]	ketone
[OHX1]-[#6]	alcohol

Morgan Matching

- An efficient way of matching multiple suitable acyclic patterns is to use a method much like Morgan’s algorithm for canonical graph labels.
- In a first pass each vertex is assigned an atom type based purely on its constitution.
- In subsequent passes, each atom’s type is updated based on its type and the types of its neighbours.

MORGAN MATCHING IN LEXICHEM

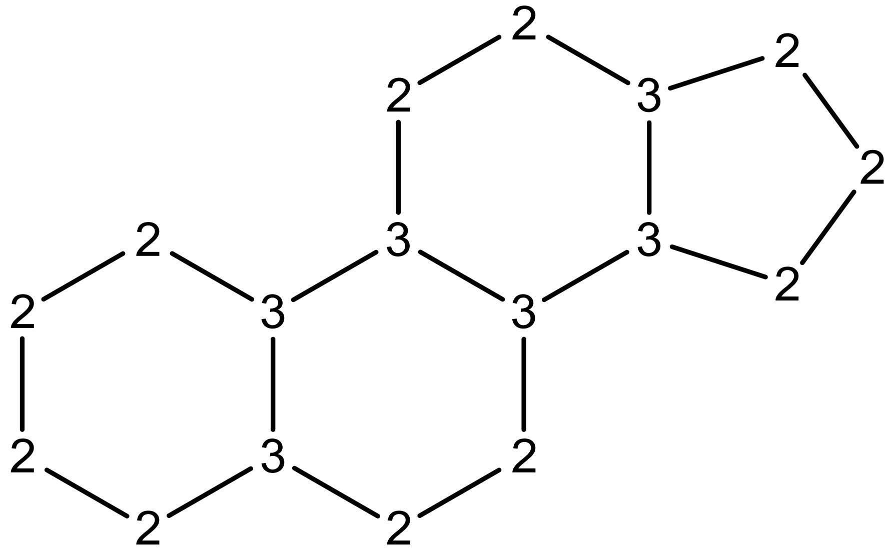
- In a first pass, atoms are assigned: 1 for [CX3], 2 for [OX1H0], 3 for [OX1H1], 4 for [NX3] and so on.
- In a second pass, the types of neighbours are used to update types: such that 1s can become 5 if they have neighbours of types 2 and 3, 6 if they have 2 and 4, and 4s can become 7s if they have a nbor of type 4.
- In a third pass, amides (type 6) can become type 8 (hydrazides) if they have a nbor of type 7 (hydrazine).

- All 1034 atom types are fully assigned in four passes.

Morgan Matching & Backtracking

- Iterative Morgan matching can work well for suitable acyclic patterns but is insufficient for ring matching.
- However, hybrid algorithms that combine MM with regular backtracking offer performance advantages.
- Consider the problem of searching for steroids.
- SMILES: C1CCC2C1CCC3C2CCC4CCCCC34
- SMARTS: [#6R2]~1~[#6R2]~[#6R2]~[#6R3]~2~[#6R3]~1~[#6R2]~[#6R2]~[#6R3]~3~[#6R3]~2~[#6R2]~[#6R2]~[#6R3]~4~[#6R2]~[#6R2]~[#6R2]~[#6R2]~[#6R3]~3~4

STEROID FIRST PASS



2x11
3x6

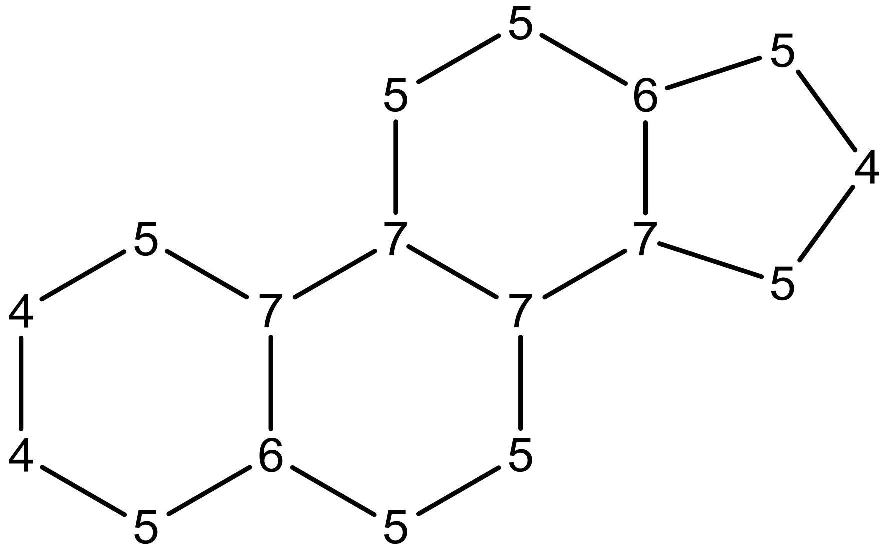
2(2,2) → 4

2(2,3) → 5

3(2,2,3) → 6

3(2,3,3) → 8

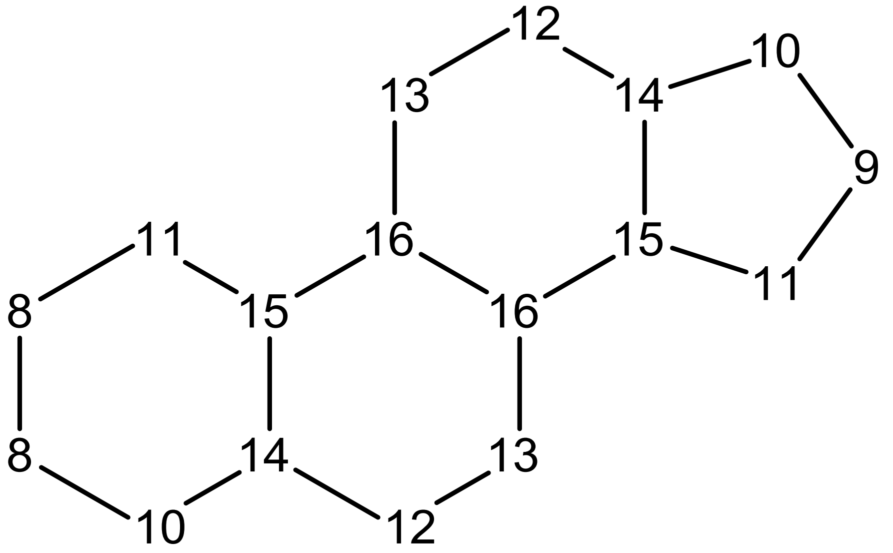
STEROID SECOND PASS



4x3
5x8
6x2
7x4

4(4,5) → 8   4(5,5) → 9
5(4,6) → 10   5(4,7) → 11   5(5,6) → 12   5(5,7) → 13
6(5,5,7) → 14
7(5,6,7) → 15   7(5,7,7) → 16

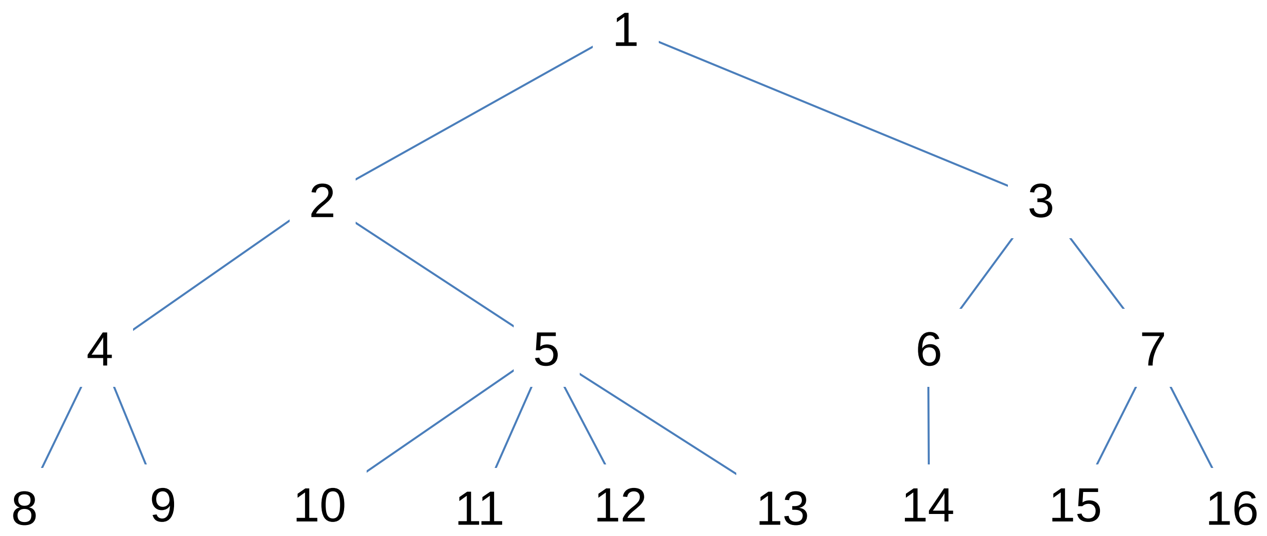
Steroid Third Pass



8x2
9x1
10x2
11x2
12x2
13x2
14x2
15x2
16x2

4(4,5) → 8	4(5,5) → 9		
5(4,6) → 10	5(4,7) → 11	5(5,6) → 12	5(5,7) → 13
6(5,5,7) → 14			
7(5,6,7) → 15	7(5,7,7) → 16		

ECFP ATOM TYPE HIERARCHY



MDL MACCS 166-BIT KEYS

- MDL's public key set of 166 substructure fragments is widely used in 2D similarity and clustering applications.
- OpenBabel, RDKit, CDK and others distribute the set of 164 SMARTS patterns corresponding to each bit of the binary fingerprint.
- For an excellent discussion on "MACCS SMARTS pattern definitions" see Andrew Dalke's post at

  http://www.mail-archive.com/rdkit-discuss@lists.sourceforge.net/msg01727.html

MDL 166-BIT MACCS KEYS

- 150 regular patterns, 8 atom pattern counts and 6 complex pattern counts.
- Relationships between patterns non-obvious
  - BIT 137 (HETEROCYCLE)   [!C;!c;R]
  - BIT 120 (HETEROCYCLIC ATOM > 1)   [!#6;R]
- Counting semantics is unusual
  - BIT 127 (A$A!O > 1)   [#8]!@*@* → [#8]!@[R]
- Monoatomic vs. polyatomic SMARTS
  - BIT 112 (AA(A)(A)A)   *~*(~*)(~*)~* → [!D0!D1!D2!D3]

    Not the same as [X4] or [D4]!

MACCS ELEMENT BITS

- 20 bits may trivially be set using a table or by "switch"ing on the atomic number of the atom
  - BITS 2 (ATOMIC NO > 103), 3 (GROUPS IVA, VA and VIA), 4 (ACTINIDES), 5 (GROUPS IIB and IVB), 6 (LANTHANIDES), 7 (GROUPS VB, VIB and VIIB), 9 (GROUP VIII), 10 (GROUP IIA), 12 (GROUP IB and IIB), 18 (GROUP IIIA), 20 (SI), 27 (I), 29 (P), 35 (GROUP IA), 42 (F), 46 (BR), 88 (S), 103 (CL), 161 (N) and 164 (O).
- But with a switch statement, BIT 134 (HALOGEN), defined as [F,Cl,Br,I] can also be handled, by setting multiple bits for some elements.

MACCS BOND BITS #1

- For asymmetric single bond SMARTS patterns it is more efficient to match the rarer side first.
  - BIT 24: [#7]-[#8] (9%) is faster than [#8]-[#7] (13%)
  - BIT 31: [F,Cl,Br,I]~[!#6!#1] (2.4%) than [!#6!#1]~[F,Cl,Br,I] (27%)
  - BIT 33: [#16]~[#7] (1.5%) is better than [#7]~[#16] (9%)
  - BIT 67: [#16]~[!#6!#1] (1.5%) than [!#6!#1]~[#16] (27%)
  - BIT 119: [#7]=* (9%) is better than *=[#7] (100%)
  - BIT 157: [#8]-[#6] (13%) is better than [#6]-[#8] (73%)
  - BIT 158: [#7]-[#6] (9%) is better than [#6]-[#7] (73%)

MACCS BOND BITS #2

- Sometimes its is more efficient to start the search from a bond rather than an atom.
- BIT 41 (CTC): [#6]#[#6]
    1. Loop over all atoms for carbons (3.9M/5.4M) then search for adjacent triple bonds (20K/9.0M).
    2. Loop over all bonds for triples (18K/5.7M) then check both ends for being carbons.
- Likewise for BIT 41 (CTN): [#6]#[#7]

To count or not to count #1

- Theoretically its is more efficient to count (single atom) patterns than repeat them multiple times in the SMARTS string.
- Consider BIT 140 (O>3): [#8].[#8].[#8].[#8]
- This should be more efficient as count([#8]) > 3.
- The first scans a molecule with 3 oxygens 16 times!

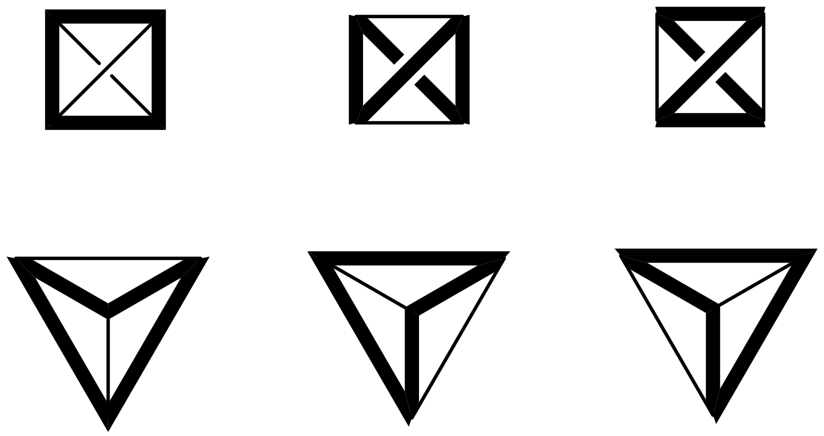
- Alas as mentioned earlier poor implementations of "count" may always allocate memory proportional to the number oxygens in the molecule (think protein!)

To Count or Not To Count #2

- For (polyatomic) SMARTS the book keeping associated with checking "unique" matches may significantly impact performance.
- However sometimes it is possible/beneficial to transform "unique" counts into "exhaustive" counts.
- BIT 130 (QQ > 1): [!#6!#1]~[!#6!#1]
- In this case, it is possible and faster to eliminate the duplicate checking for ucount([!#6!#1]~[!#6!#1]) > 1 and instead test for count([!#6!#1]~[!#6!#1]) > 2.

UNIQUE ATOMS VS UNIQUE BONDS

- Most subgraph isomorphism implementations provide a method of returning back matches that are atom unique.
- However there are applications that require the set of bond unique subgraphs, which isn't the same.



MACCS RING BITS

- A significant fraction of time is spent on the ring bits
  - BITS 22 (3M RING), 11 (4M RING), 96 (5M RING), 163 (6M RING), 19 (7M RING) and 101 (8M RING).
- How a ring SMARTS is expressed affects performance

– *~1~*~*~*~1	Poor
– *@1@*@*@*@1	Good
– [R]~1~[R]~[R]~[R]~1	Better
– [R]@1@*@*@*@1	Best

- However the most significant improvements come from matching these ring patterns simultaneously.

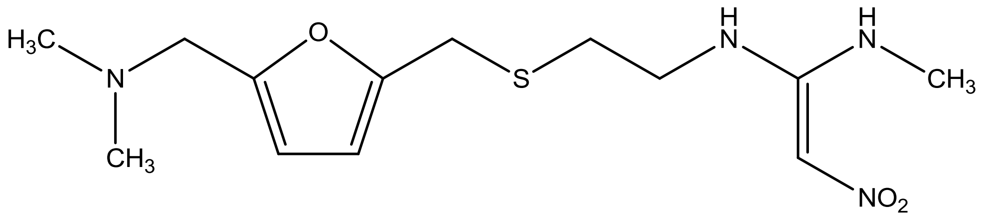
MACCS FINGERPRINTING TIMES

Implementation	Total	Match
File I/O	9.02s	
BIT 125 (AROM > 1)	9.54s	0.52s
Original SMARTS	226.21s	217.19s
OpenEye Impl	216.11s	207.09s
Generated Impl	115.37s	106.28s
Speed-up	1.96x	2.05x

Derwent CPI Codes

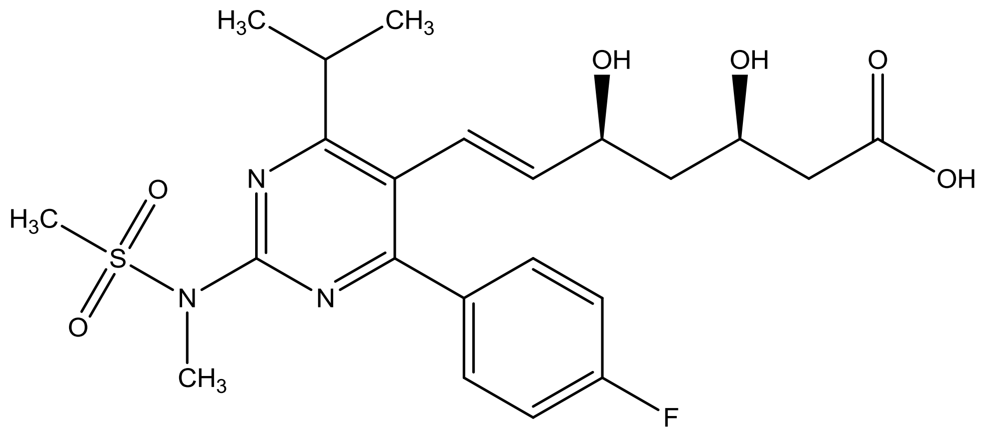
- A far more complex “fragment fingerprint” are the CPI codes used by Derwent/Thompson Reuters to index the “World Patent Index”.
- These codes form the basis of WPI structure searching in STNExpress.
- CPI uses about 1080 substructure/fragment codes that can be assigned automatically.

CPI Indexing Example: Ranitidine



F012 – Mono-heterocycle substituted at 2-position
F015 – Mono-heterocycle substituted at 5-position
F111 – 5-Membered mono-heterocycle with 1 oxygen: C_4H_40; furan
H102 – Secondary amine (N-atom not in a ring)
H103 – Tertiary amine (N-atom not in a ring)
H183 – > 2 Amino groups bonded to C-atom of aliphatic group
H381 – 1 Nitro group bonded to non-cyclic (aliphatic) C-atom
H598 – S-atom of thioether group bonded to C-atom of acyclic group (no ring atoms)
H721 – 1 Unconjugated acyclic C=C
L640 – N-C-U group (U is N, O, S, Se or Te)
M211 – Methane or methyl group
M413 – Organic with >=1 unfused heterocycles, but no fused heterocycles
M521 – Organic containing 1 unfused heterocyclic ring
M540 – Organic containing no non-aromatic carbocyclic rings.

CPI INDEXING EXAMPLE: ROSUVASTATIN



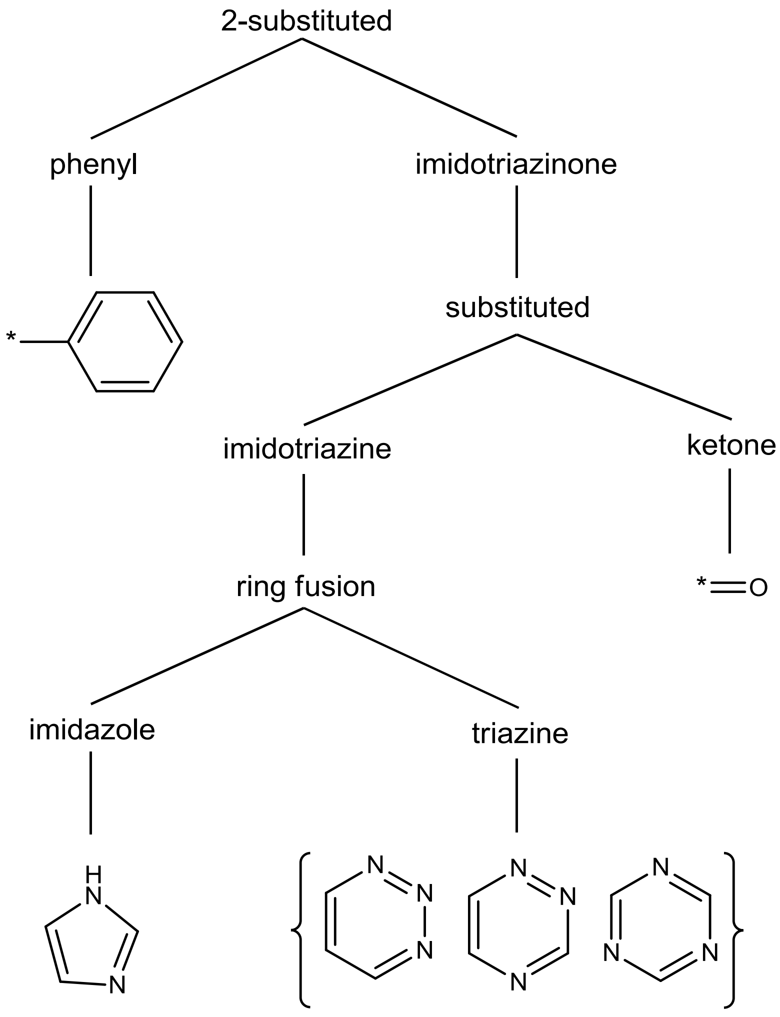
C316 – Inorganic sulphur, selenium or tellurium present with valency > 5
F012 – Mono-heterocycle substituted at 2-position
F541 – 6-membered mono-heterocycle with 2N; C_4H_4N2 (pyrimidine)
G013 – 2 positions substituted on unfused benzene; 1,4- (para)
G100 – 1 or more uncondensed benzenes with no other carbocycles
H482 – 2 -OH groups bonded to C-atoms of acyclic groups
H601 – Fluorine present [except in C(=[O,S])X or trifluoromethyl]
H641 – 1 Halogen bonded to carbocyclic aromatic ring
J011 – Total number of carboxylic acids, esters and amides = 1
J171 – 1 C(=O)OH group bonded to C-atom of acyclic group or formic acid
K353 – Other organic S-N groups

PATENT TEXT MINING

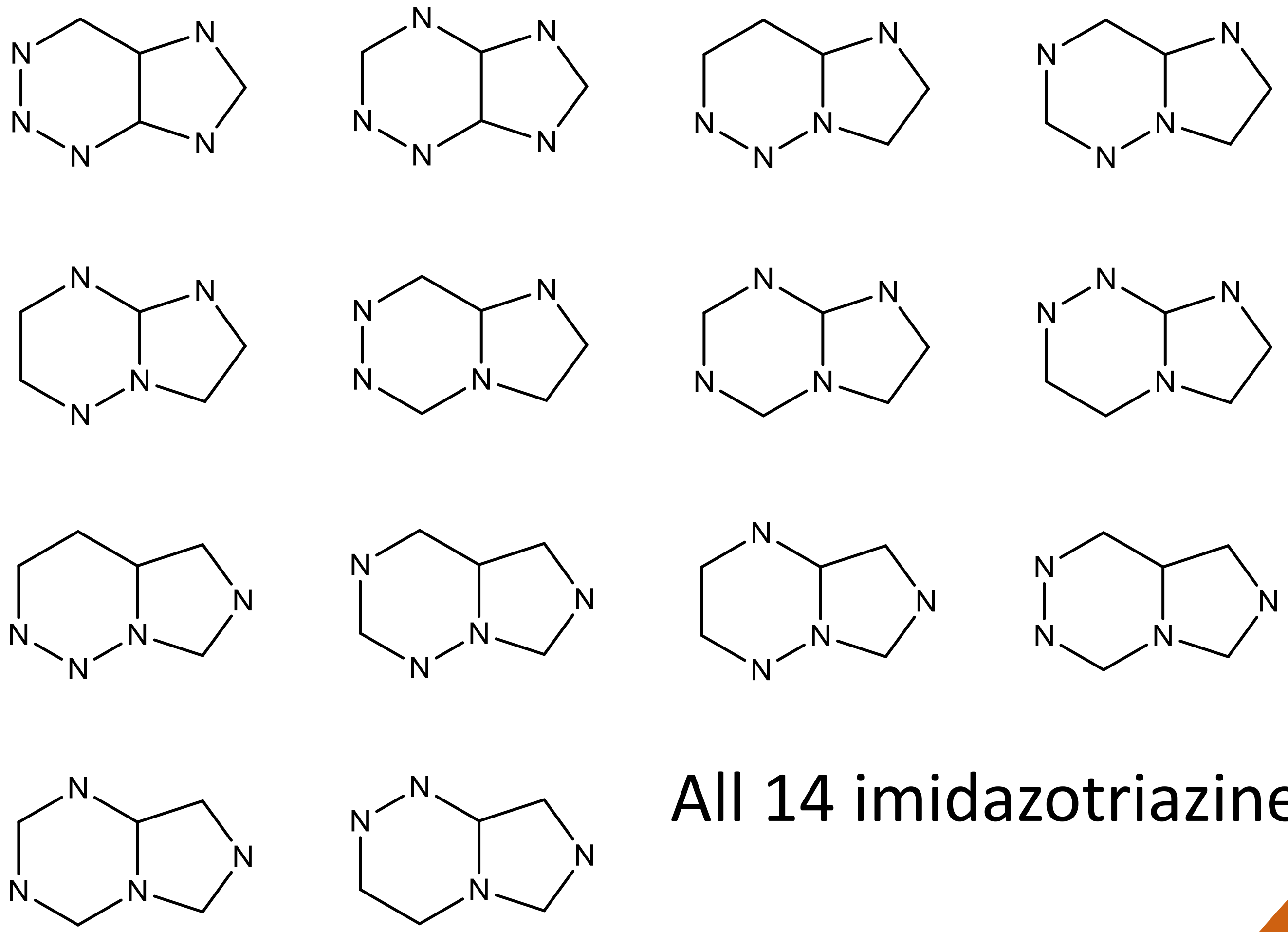
- The classic example of pharmaceutical patent busting is the 2009 Bayer patent for Vardenafil (Levitra), entitled "**2-phenyl substituted imidazotriazinones as phosphodiesterase inhibitors**", US patent number 7,696,206(B2).

- How much information can/could be mined from the title alone?

2-PHENYL SUBSTITUTED TRIAZINONES

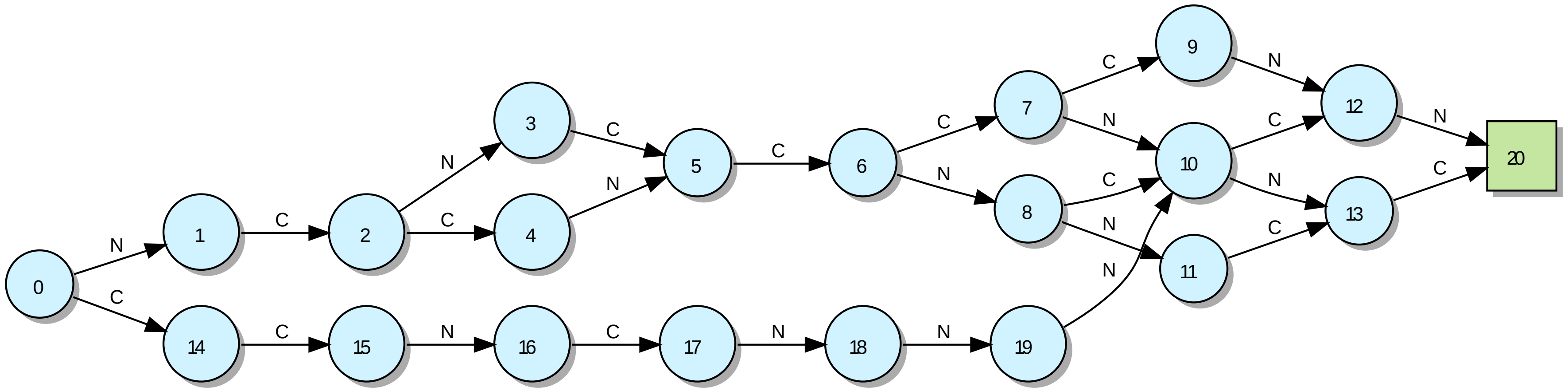


IMIDAZOTRIAZINES



All 14 imidazotriazines

BACKTRACKING STATE MACHINE



Simultaneously matching all 14 imidazotriazine SMARTS can be accomplished by the above finite state machine (FSM). A single SMARTS match is conceptually a linear state sequence.

CONCLUSIONS

- The “compilation” of SMARTS patterns, Patsy, is shown to dramatically improve the run-time performance of subgraph matching.
- The biggest gains are seen when matching multiple patterns, as compilation allows some patterns to be matched simultaneously.
- Increasing speed leads to improvements in expressive power, enabling previously impossible/prohibitive applications/queries.

APPLICATIONS/FUTURE WORK

- Reaction Informatics
  - Matching of a potentially large number of transformations for both normalization and synthetic/retrosynthetic analysis.
- Markush Representation
  - Capturing the semantics/scope of pharmaceutical patent claims.

THANK YOU FOR YOUR TIME

- Thanks again to the ICCS organisers.
- And thanks for your questions.