Electronic Supplementary Information (ESI)

Chemistry in a graph: Modern Insights into Commercial Organic Synthesis Planning

Authors:

Claudio Avila (ORCID 0000-0002-5691-6814), Adam West (ORCID 0009-0006-0061-1495), Anna Vicini (ORCID 0009-0009-5051-1523), William Waddington (ORCID 0009-0006-6496-0301), Christopher Brearley (ORCID 0009-0006-1744-3272), James Clarke (ORCID 0009-0008-4111-2427), Andrew Derrick (ORCID 0000-0002-0482-9057)

Affiliation:

Pfizer UK R&D Ltd, Pharmaceutical Sciences Small Molecule (PSSM), Chemical Research and Development (CRD).

Address:

Discovery Park House, Ramsgate Road, Sandwich (United Kingdom), CT13 9NJ

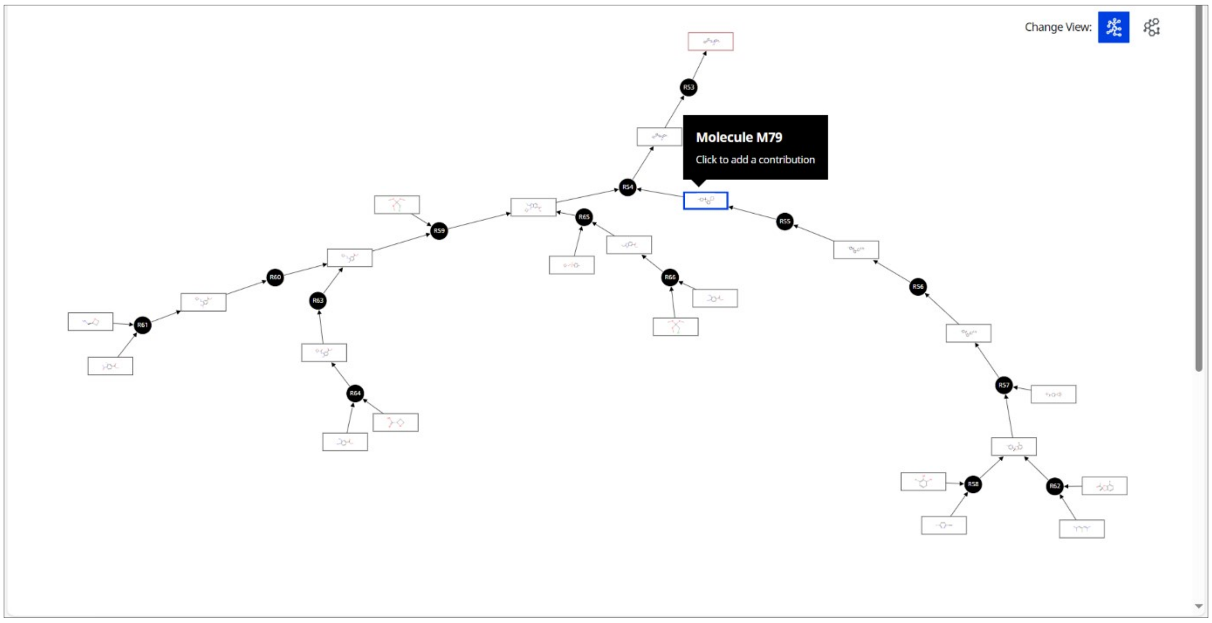
Contents

ESI-1 – Digitisation of human generated ideas (digitalised brainstorm)

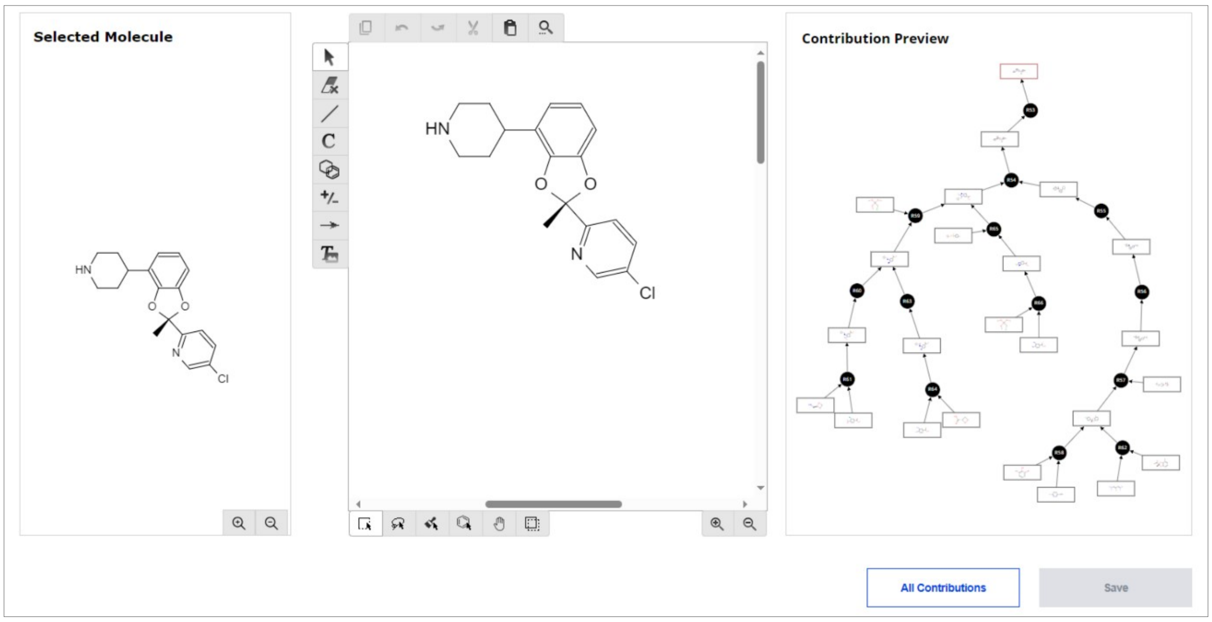
1.1 Target molecule properties

Synonyms	PF-07081532, Lotiglipron
SMILES	C[C@@]1(OC2=CC=CC(=C2O1)C3CCN(CC3)CC4=NC5=C(N4C[C@@H]6CCO6)C=C(C=C5)C(=O)O)C7=NC=C(C=C7)Cl
INCHI	InChI=1S/C31H31ClN4O5/c1-31(27-8-6-21(32)16-33-27)40-26-4-2-3-23(29(26)41-31)19-9-12-35(13-10-19)18-28-34-24-7-5-20(30(37)38)15-25(24)36(28)17-22-11-14-39-22/h2-8,15-16,19,22H,9-14,17-18H2,1H3,(H,37,38)/t22-,31-/m0/s1
INCHI Key	SVPYZAJTWFQTSM-UGDMGKLASA-N

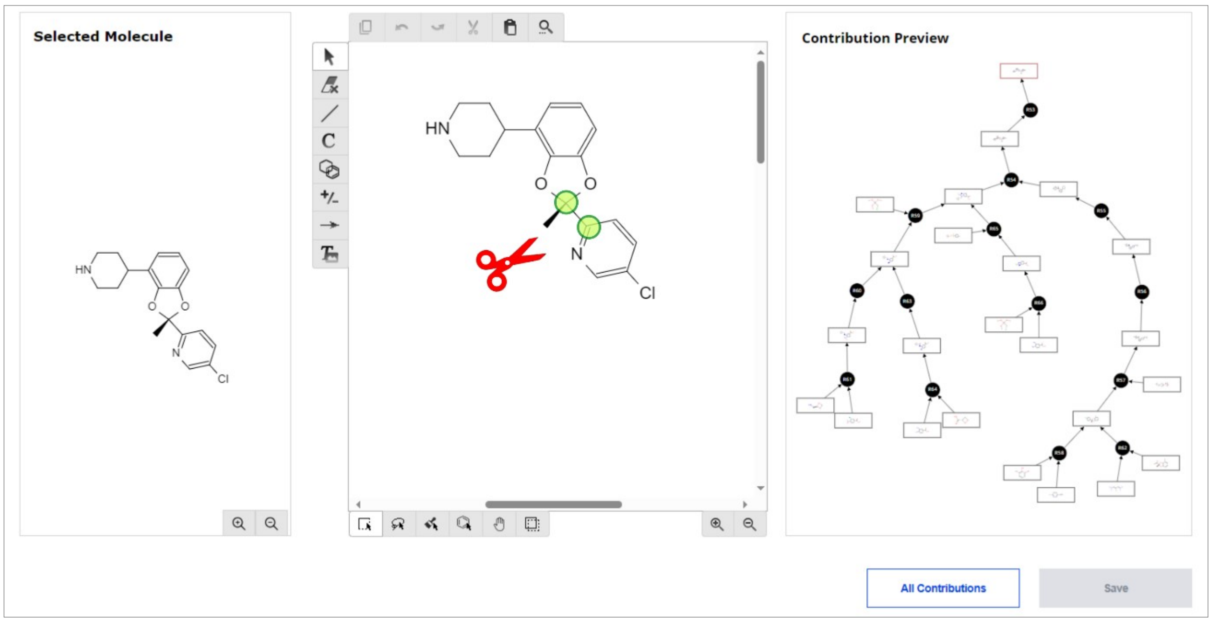
1.2 Initial network where contribution (disconnection) is made, and selection of the specific molecule node to be expanded. Specific molecule details for the entire network are provided in section 2.4.



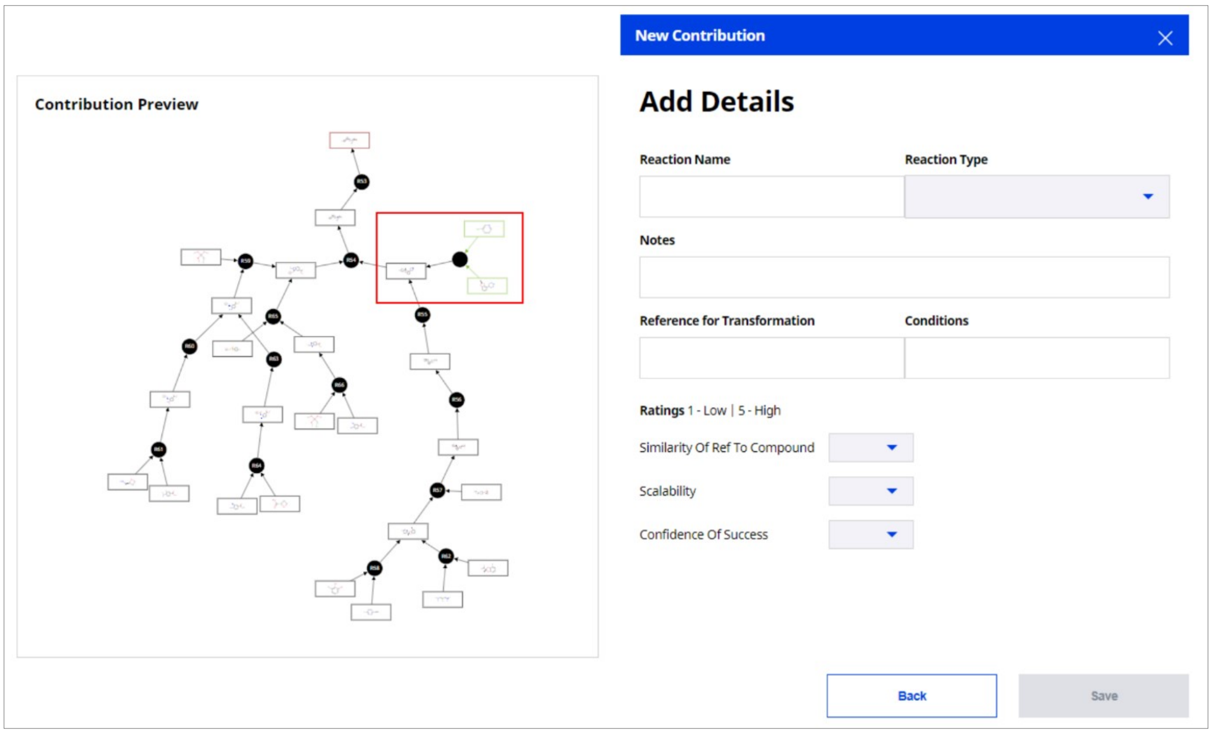
1.3 Initial rendering of the molecule, with a digital user interface for a chemist to draw the transformation (centre of the picture)



1.4 Resulting chemical transformation to be introduced, generating two individual fragments from the initial selected molecule. This could include leaving groups or only main fragments.



1.5 Summary of the contribution before submitting an entry. Metadata can be added considering chemist or contributor experiences. This can include background information searches to autocomplete information, such as literature refences, safety data from modelling or published, etc. When this step is ready, data is written in the graph database supporting the defined data model (Figure 2). Form is displayed alongside to capture additional information, with fields aligned to SELECT criteria (or any other standard used).



Add Details

Reaction Name

Reaction Type

Notes

Reference for Transformation

Conditions

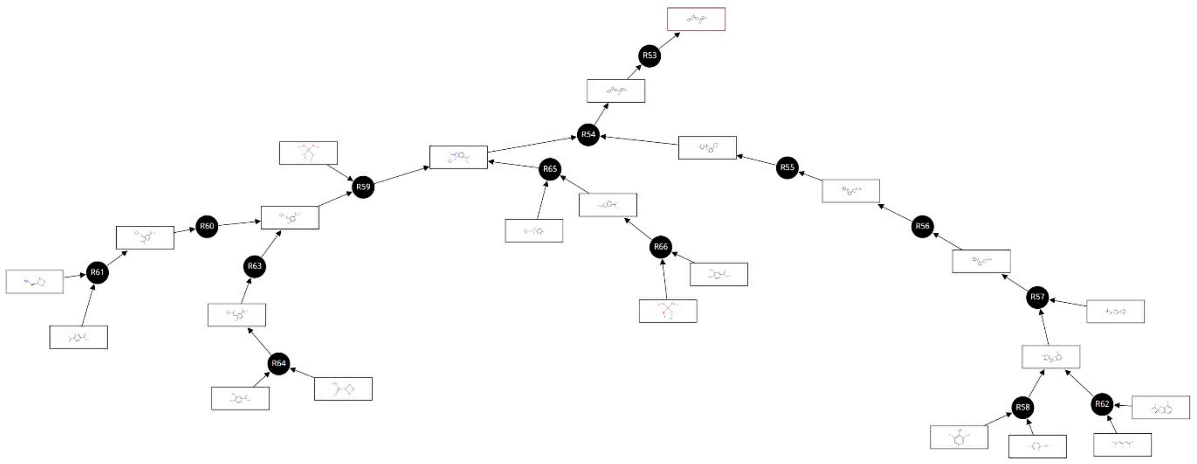
Ratings 1 - Low | 5 - High

Similarity Of Ref To Compound

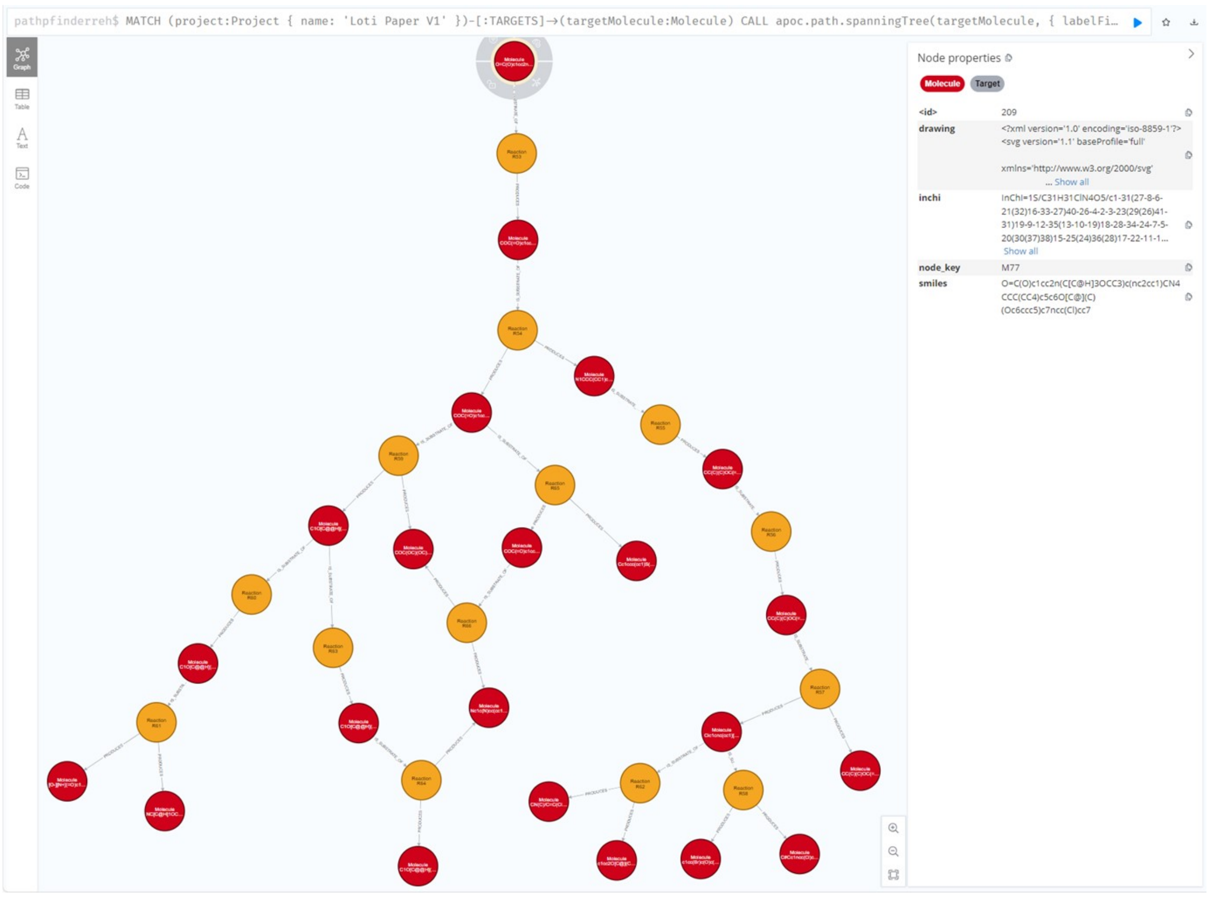
Scalability

Confidence Of Success

1.6 Network view with ideas already included into the graph database and accessible to anyone.



1.7 Same network shown in 1.6, visualised from the graph database directly (Neo4J backend). Original image above, image with indications below. Note the rendering differences. As molecules are unique on the graph database, the substrate feeding R59 and R66 appear only once in the database (same as substrate feeding R64 and R66). This creates a recurrence within the graph (see second image below with indications).

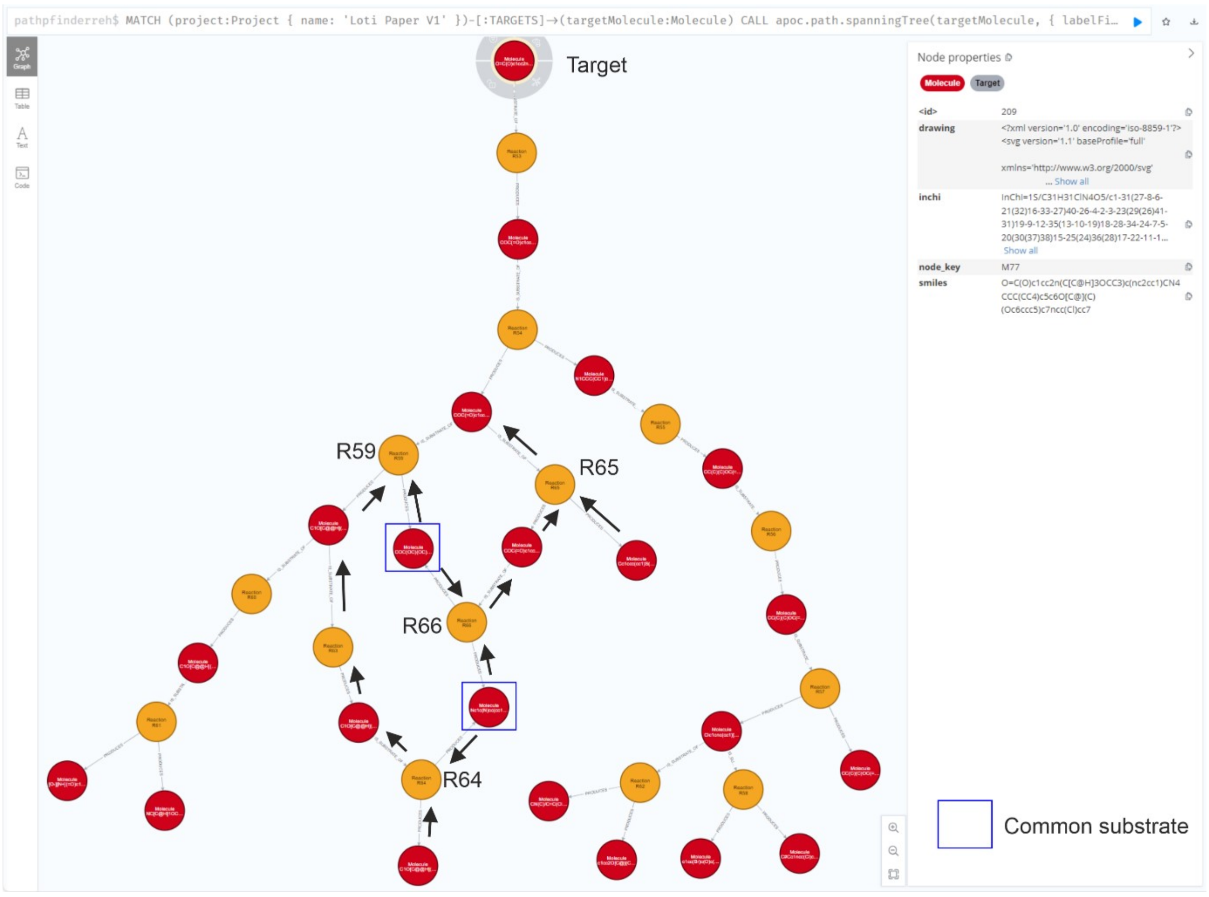


```
pathpfinderreh$ MATCH (project:Project { name: 'Loti Paper V1' })-[:TARGETS]→(targetMolecule:Molecule) CALL apoc.path.spanningTree(targetMolecule, { labelFi…
```

Node properties

Molecule Target

<id>	209
drawing	<?xml version='1.0' encoding='iso-8859-1'?> <svg version='1.1' baseProfile='full' xmlns='http://www.w3.org/2000/svg' ... Show all
inchi	InChI=1S/C31H31ClN4O5/c1-31(27-8-6-21(32)16-33-27)40-26-4-2-3-23(29(26)41-31)19-9-12-35(13-10-19)18-28-34-24-7-5-20(30(37)38)15-25(24)36(28)17-22-11-1... Show all
node_key	M77
smiles	O=C(O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC4)c5c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7



```
pathpfinderreh$ MATCH (project:Project { name: 'Loti Paper V1' })-[:TARGETS]→(targetMolecule:Molecule) CALL apoc.path.spanningTree(targetMolecule, { labelFi…
```

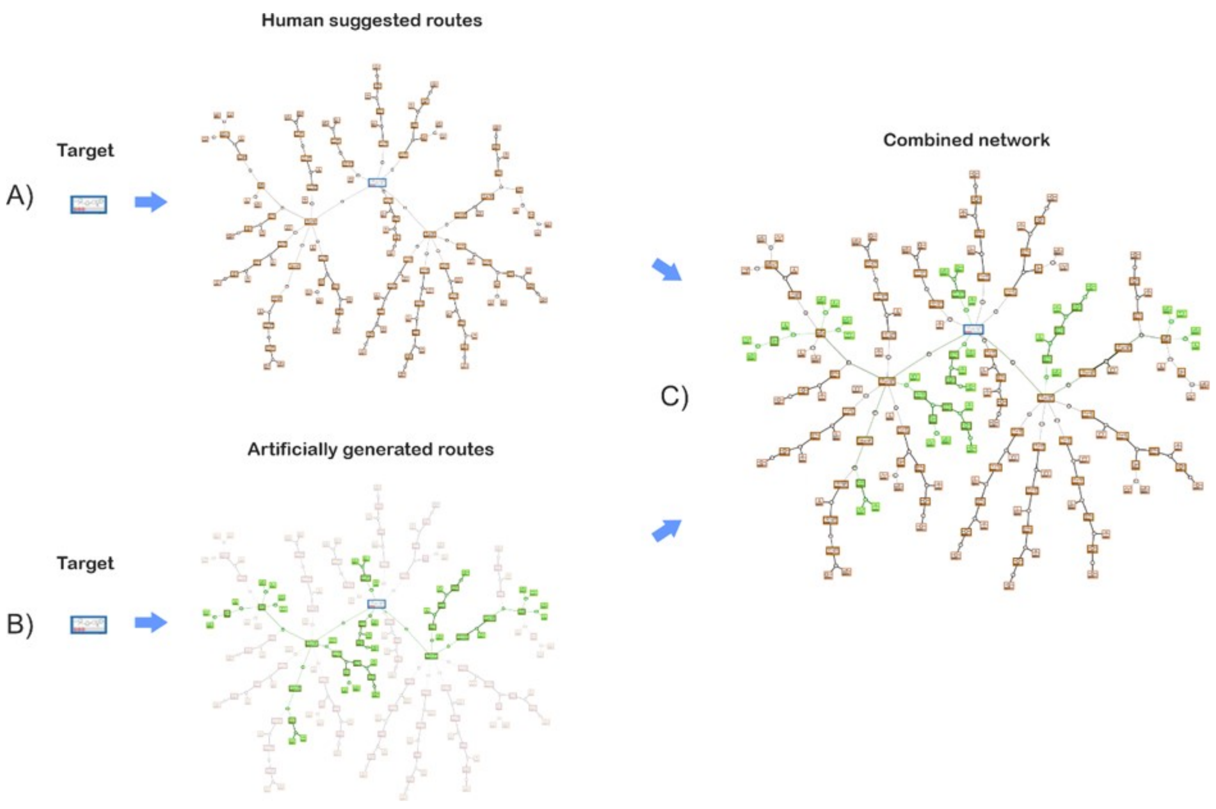
Node properties

Molecule Target

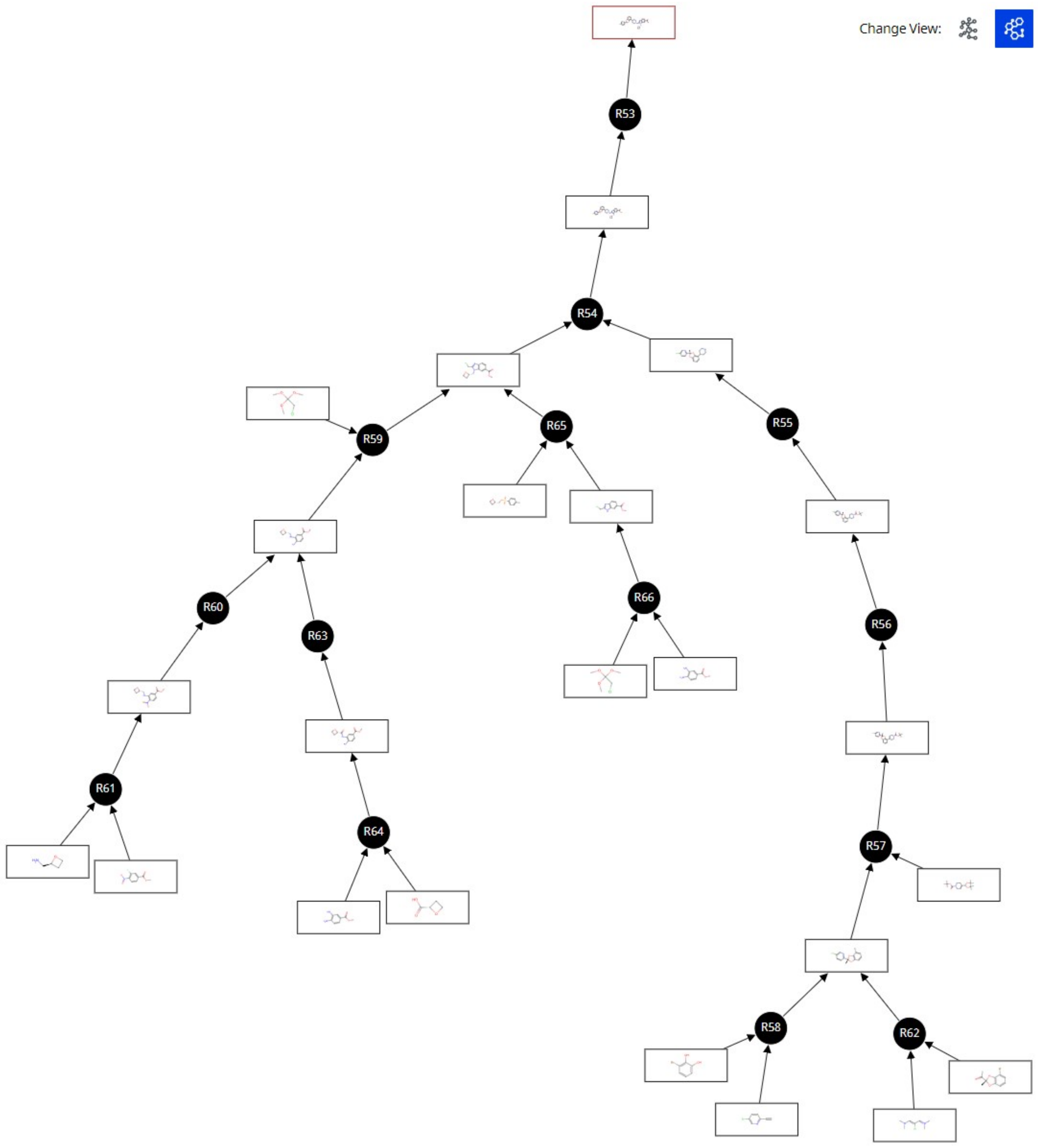
<id>	209
drawing	<?xml version='1.0' encoding='iso-8859-1'?> <svg version='1.1' baseProfile='full' xmlns='http://www.w3.org/2000/svg' ... Show all
inchi	InChI=1S/C31H31ClN4O5/c1-31(27-8-6-21(32)16-33-27)40-26-4-2-3-23(29(26)41-31)19-9-12-35(13-10-19)18-28-34-24-7-5-20(30(37)38)15-25(24)36(28)17-22-11-1... Show all
node_key	M77
smiles	O=C(O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC4)c5c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7

ESI-2 – Synthesis of ***Lotiglipron PF-07081532*** – full network produced.

2.1 Combining human with artificially generated routes – a theoretical network example is shown below. The same procedure was implemented for a real molecule, and it is described in the next sections.



2.2 For the target molecule (Lotiglipron), some of the routes generated by human-led retrosynthesis analysis (following the procedure described in ESI-1) are shown in the image below:

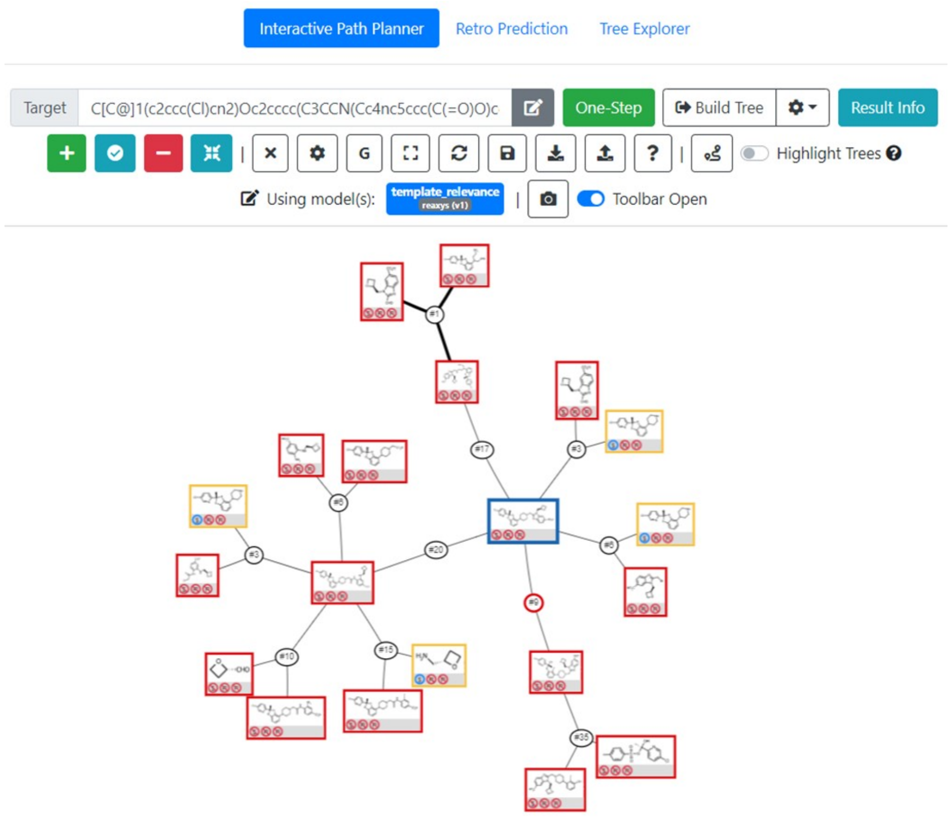


The equivalent graph stored in the graph database is also shown below:

2.3 Some retrosynthesis routes generated by assisting predictive software (ASKCOS)

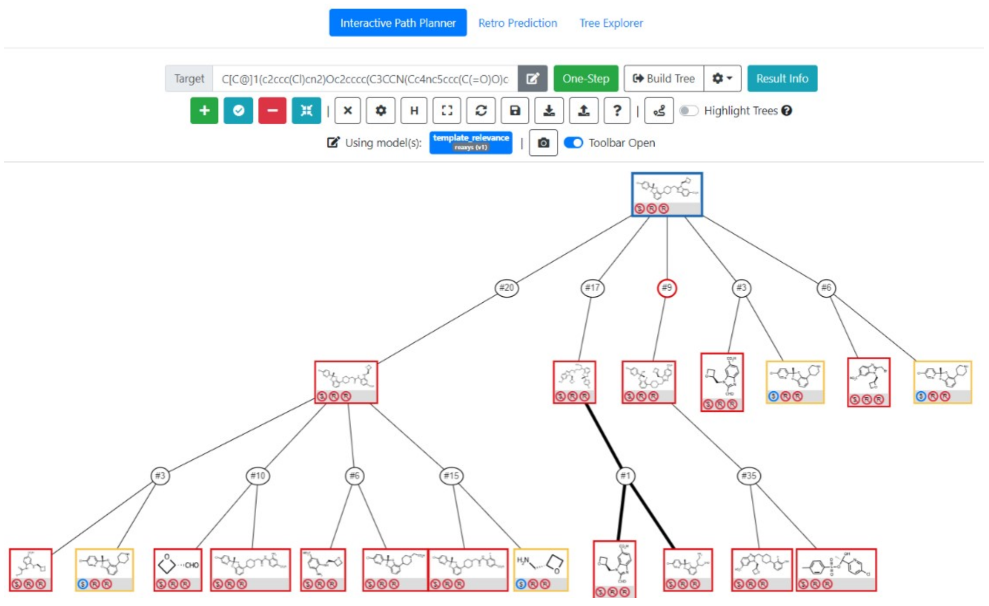
Details of all the routes digitally suggested are presented in section 3 (ESI-3). A summary figure with the routes filtered from ASCKOS is presented in the graph below.

Network view:


Target: C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(Cc4nc5ccc(C(=O)O)c

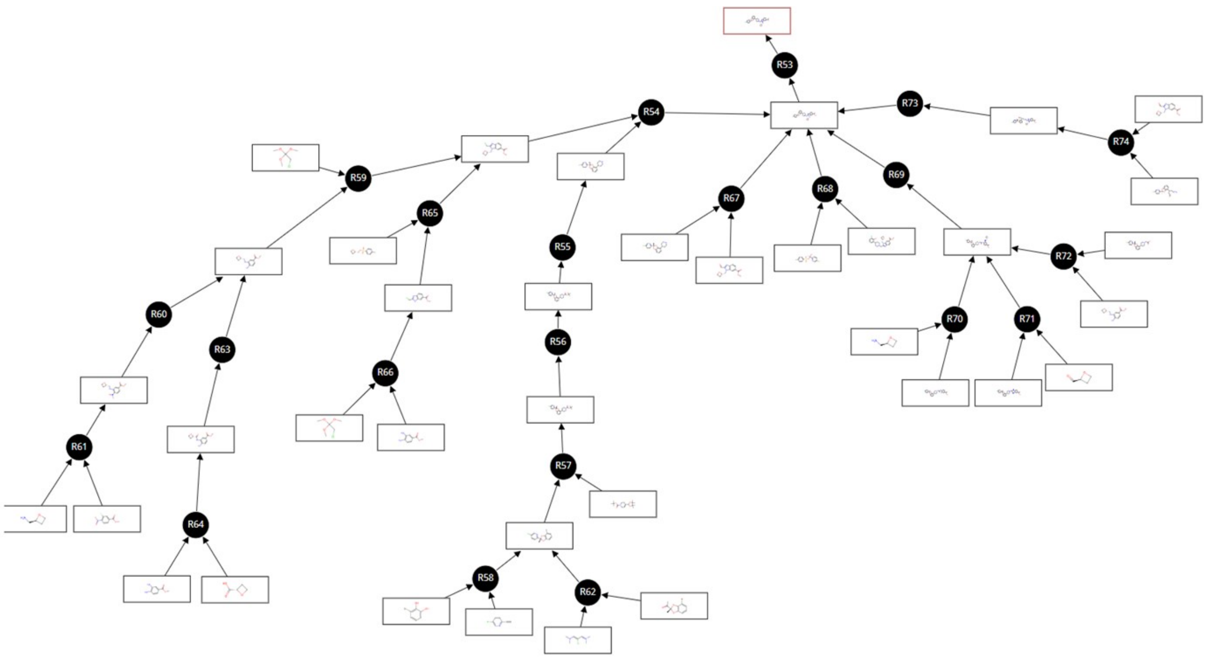
Using model(s): template_relevance reaxys (v1)

Alternative tree view:

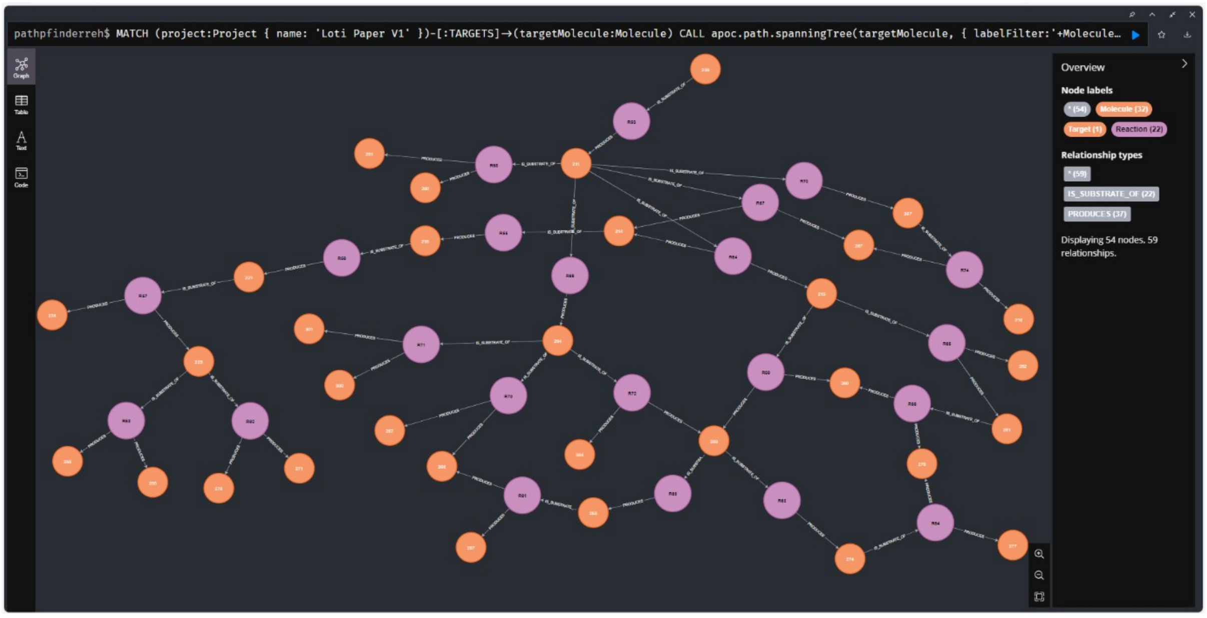

Target: C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(Cc4nc5ccc(C(=O)O)c

Using model(s): template_relevance reaxys (v1)

2.4 Combined network – merging human with synthetically generated ideas for Lotiglipron



The corresponding graph database representation (native Neo4J) is also shown below:



```
pathpfinderreh$ MATCH (project:Project { name: 'Loti Paper V1' })-[:TARGETS]→(targetMolecule:Molecule) CALL apoc.path.spanningTree(targetMolecule, { labelFilter:'+Molecule…
```

Overview

Node labels

* (54) Molecule (32) Target (1) Reaction (22)

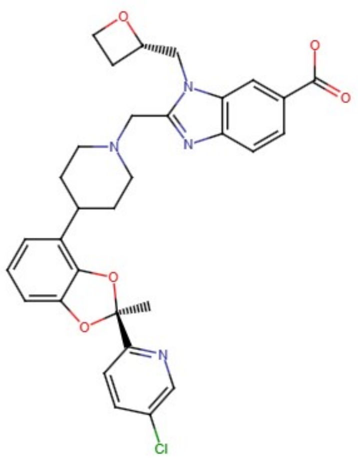
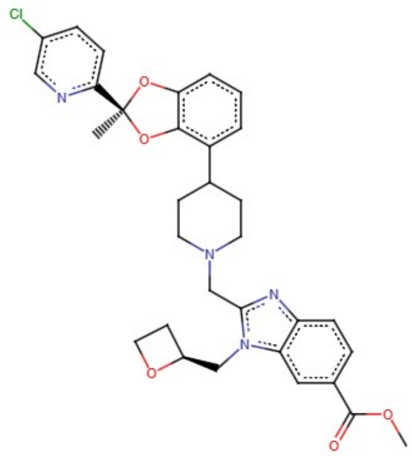
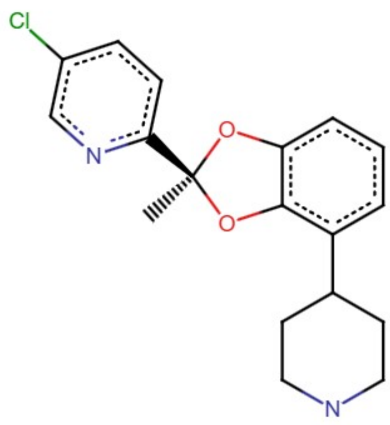
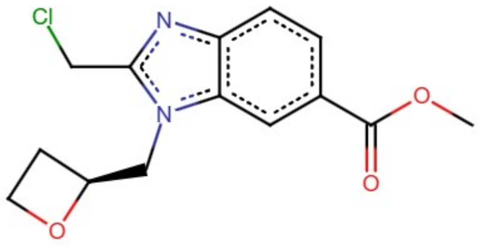
Relationship types

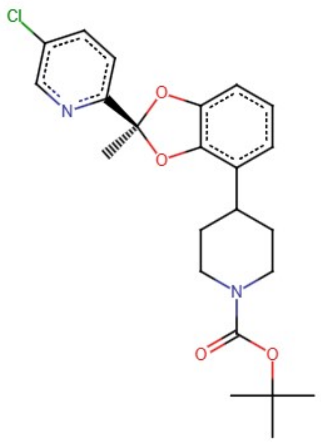
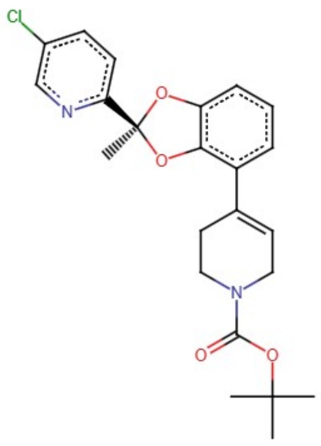
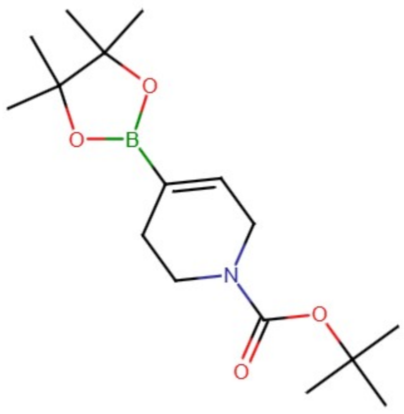
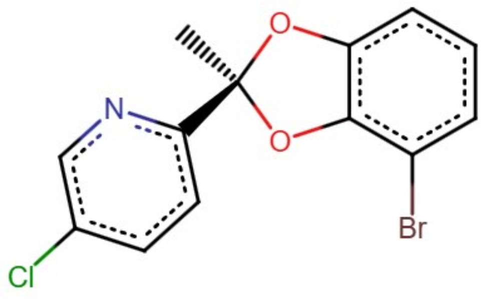
* (59) IS_SUBSTRATE_OF (22) PRODUCES (37)

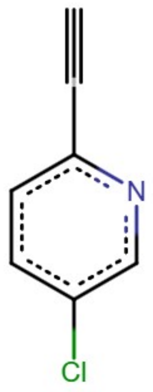
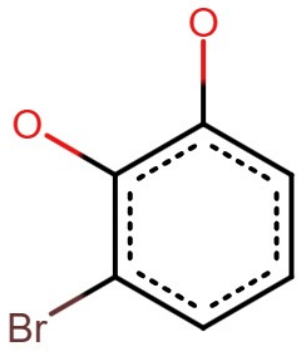
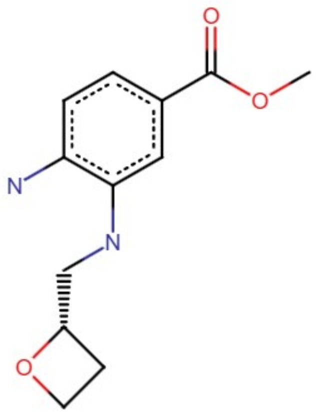
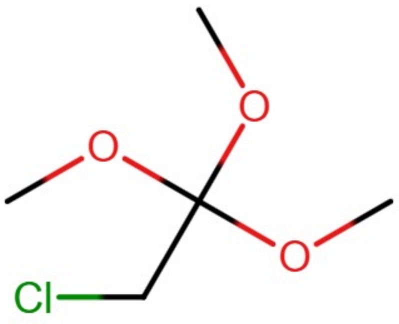
Displaying 54 nodes, 59 relationships.

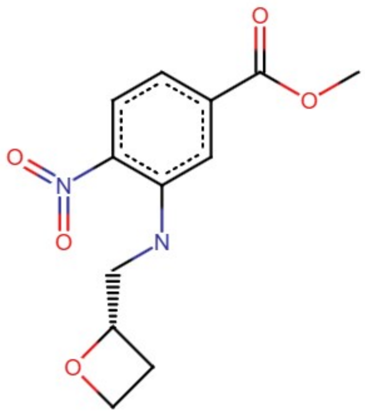
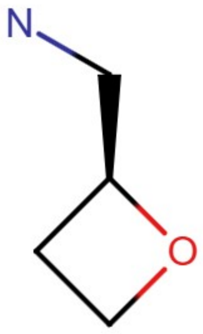
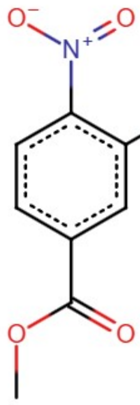
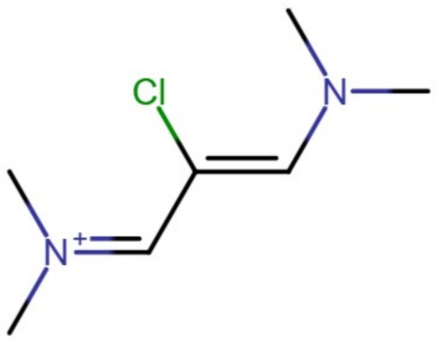
Specific molecule details are provided in the table below:

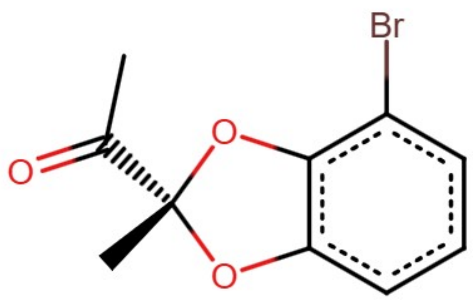
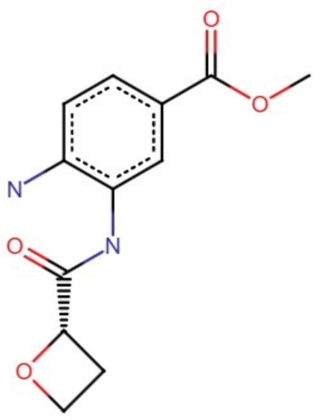
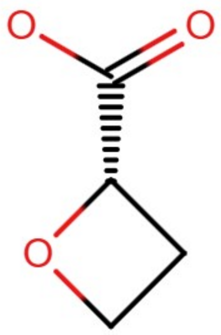
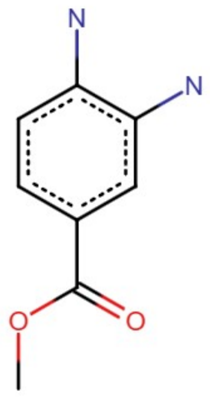
Molecule ID (picture display)	Node_Key Neo4j	Smiles	

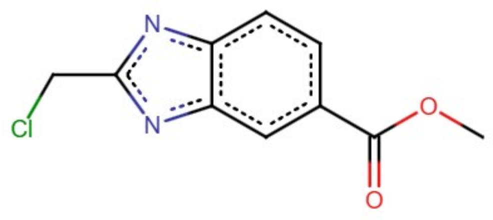
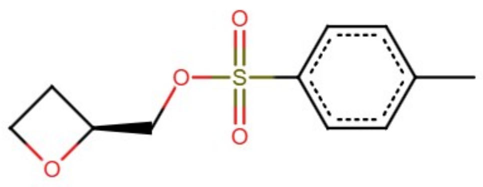
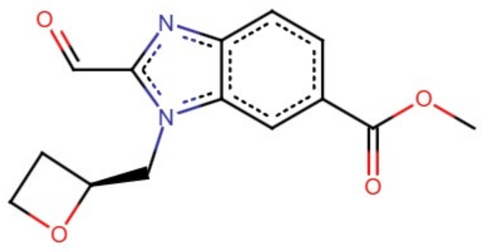
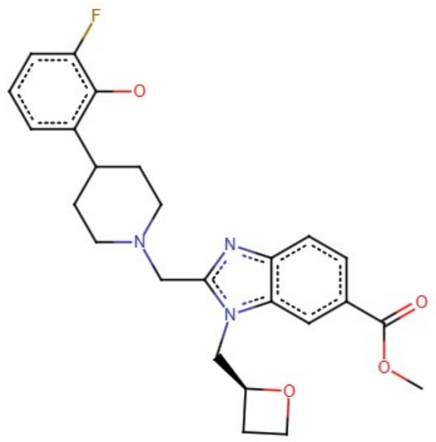
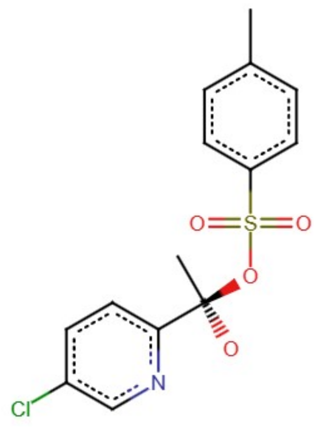
209	M77	C[C@@]1(OC2=CC=CC(=C2O1)C3CCN(CC3)CC4=NC5=C(N4C[C@@H]6CCO6)C=C(C=C5)C(=O)O)C7=NC=C(C=C7)Cl	
211	M78	COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC4)c5c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7	
214	M79	N1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4	
215	M80	COC(=O)c1cc2n(C[C@H]3OCC3)c(CCl)nc2cc1	

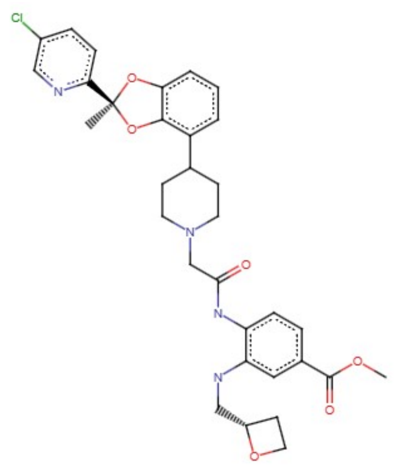
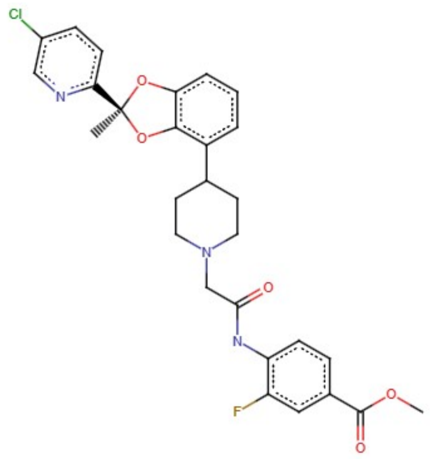
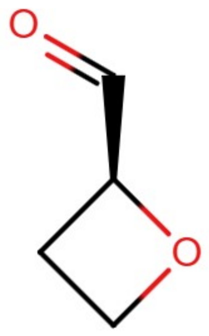
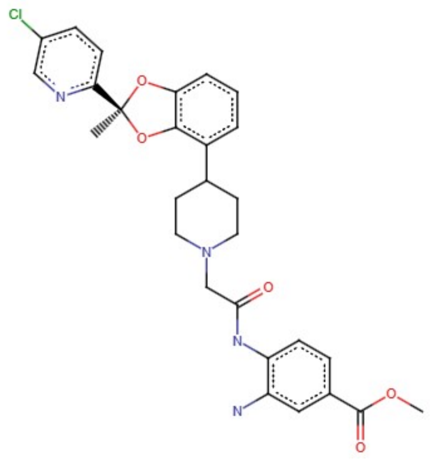
218	M81	CC(C)(C)OC(=O)N1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4	
221	M82	CC(C)(C)OC(=O)N1CCC(=CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4	
224	M84	CC(C)(C)OC(=O)N1CCC(=CC1)B2OC(C)(C)C(C)(O2)C	
225	M83	Clc1cnc(cc1)[C@]2(C)Oc3c(O2)cccc3Br	

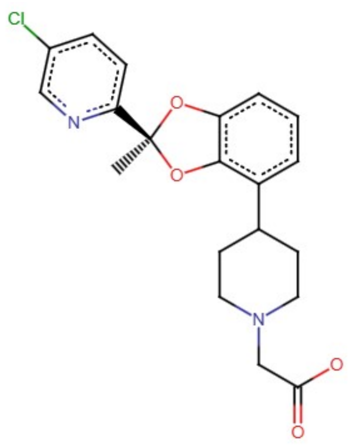
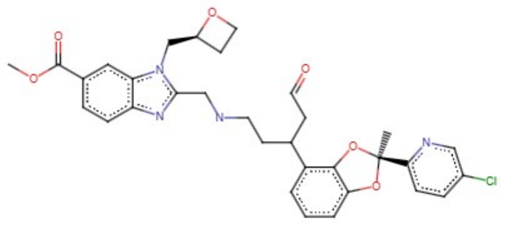
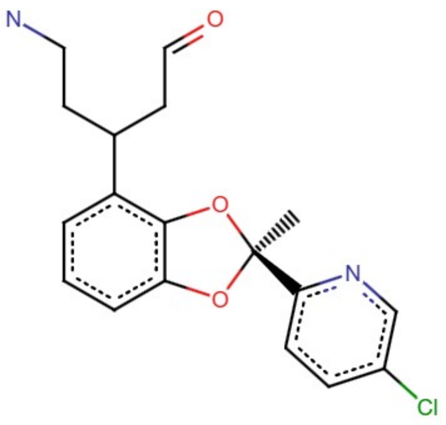
255	M85	C#Cc1ncc(Cl)cc1	
256	M86	c1cc(Br)c(O)c(c1)O	
259	M87	C1O[C@@H](C1)CNc2c(N)ccc(c2)C(=O)OC	
260	M88	COC(OC)(OC)CCl	

263	M89	C1O[C@@H](C1)CNc2c(N(=O)=O)ccc(c2)C(=O)OC	
266	M90	NC[C@H]1OCC1	
267	M91	[O-][N+](=O)c1c(F)cc(cc1)C(=O)OC	
270	M92	CN(C)/C=C(Cl)/C=[N+](C)C	

271	M93	c1cc2O[C@](C)(C(=O)C)Oc2c(c1)Br	
274	M94	C1O[C@@H](C1)C(=O)Nc2c(N)ccc(c2)C(=O)OC	
277	M95	C1O[C@@H](C1)C(O)=O	
278	M96	Nc1c(N)cc(cc1)C(=O)OC	

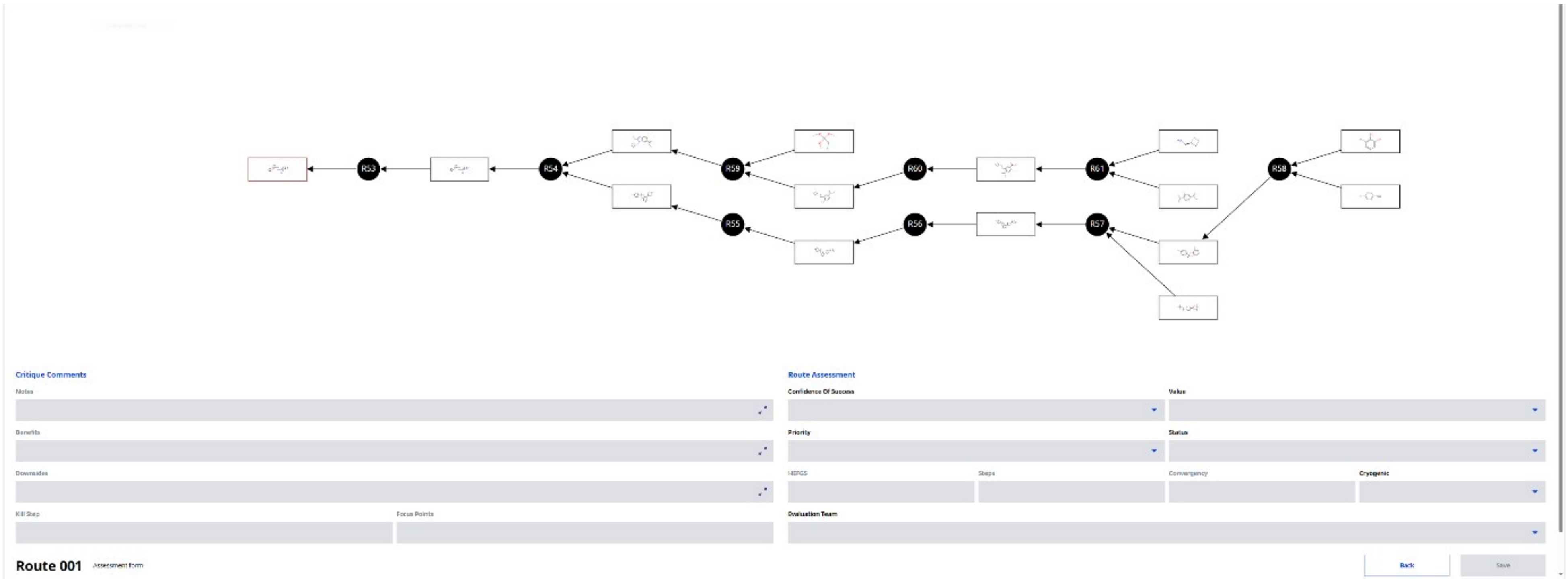
281	M97	COC(=O)c1cc2c(cc1)nc([nH]2)CCl	
282	M98	Cc1ccc(cc1)S(=O)(=O)OC[C@H]2OCC2	
287	M99	COC(=O)c1cc2n(C[C@H]3OCC3)c(C=O)nc2cc1	
290	M100	Fc1c(O)c(ccc1)C2CCN(CC2)Cc3n(C[C@H]4OCC4)c5c(n3)ccc(c5)C(=O)OC	
291	M101	Cc1ccc(cc1)S(=O)(=O)O[C@@](C)(O)c2ncc(Cl)cc2	

294	M102	C1O[C@@H](C1)CNc2c(ccc(c2)C(OC)=O)NC(=O)CN3CCC(CC3)c4c5O[C@](C)(Oc5ccc4)c6ncc(Cl)cc6	
297	M103	COC(=O)c1cc(F)c(cc1)NC(=O)CN2CCC(CC2)c3c4O[C@](C)(Oc4ccc3)c5ncc(Cl)cc5	
300	M104	O=C[C@H]1OCC1	
301	M105	COC(=O)c1cc(N)c(cc1)NC(=O)CN2CCC(CC2)c3c4O[C@](C)(Oc4ccc3)c5ncc(Cl)cc5	

304	M106	O=C(O)CN1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4	
307	M107	COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CNCCC(CC=O)c4c5O[C@](C)(Oc5ccc4)c6ncc(Cl)cc6	
310	M108	NCCC(CC=O)c1c2O[C@](C)(Oc2ccc1)c3ncc(Cl)cc3	

2.5 Individual routes identified.

An algorithm transversed the network automatically identifying 12 routes. Details are shown below and as seen directly from the user interface developed. A form is displayed alongside each route to capture route data and aligned to the SELECT criteria.



Critique Comments

Notes

Benefits

Downsides

Kill Step

Focus Points

Route Assessment

Confidence Of Success

Value

Priority

Status

HSFGS

Steps

Convergency

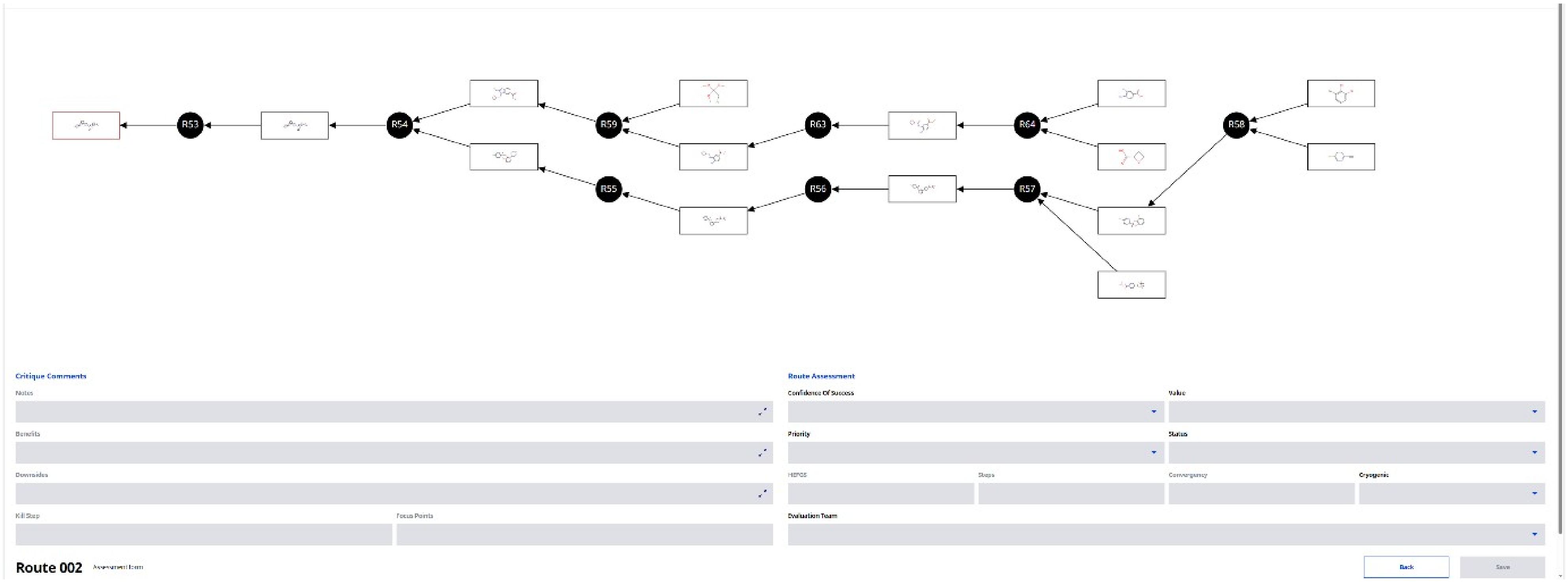
Cryogenic

Evaluation Team

Route 001 Assessment form

Back

Save



Critique Comments

Notes

Benefits

Downsides

Kill Step

Focus Points

Route Assessment

Confidence Of Success

Value

Priority

Status

HEFGS

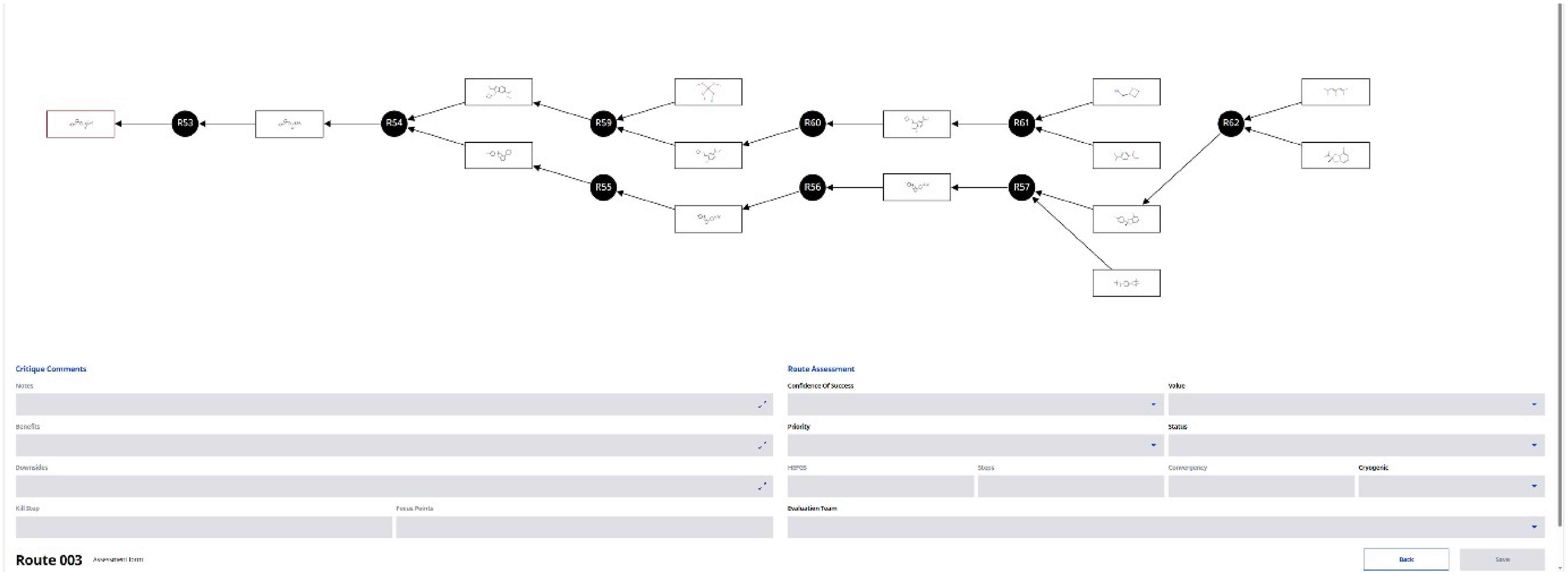
Steps

Convergency

Cryogenic

Evaluation Team

Route 002 Assessment form



Critique Comments

Notes

Benefits

Downsides

Kill Step

Focus Points

Route Assessment

Confidence Of Success

Value

Priority

Status

HIPOS

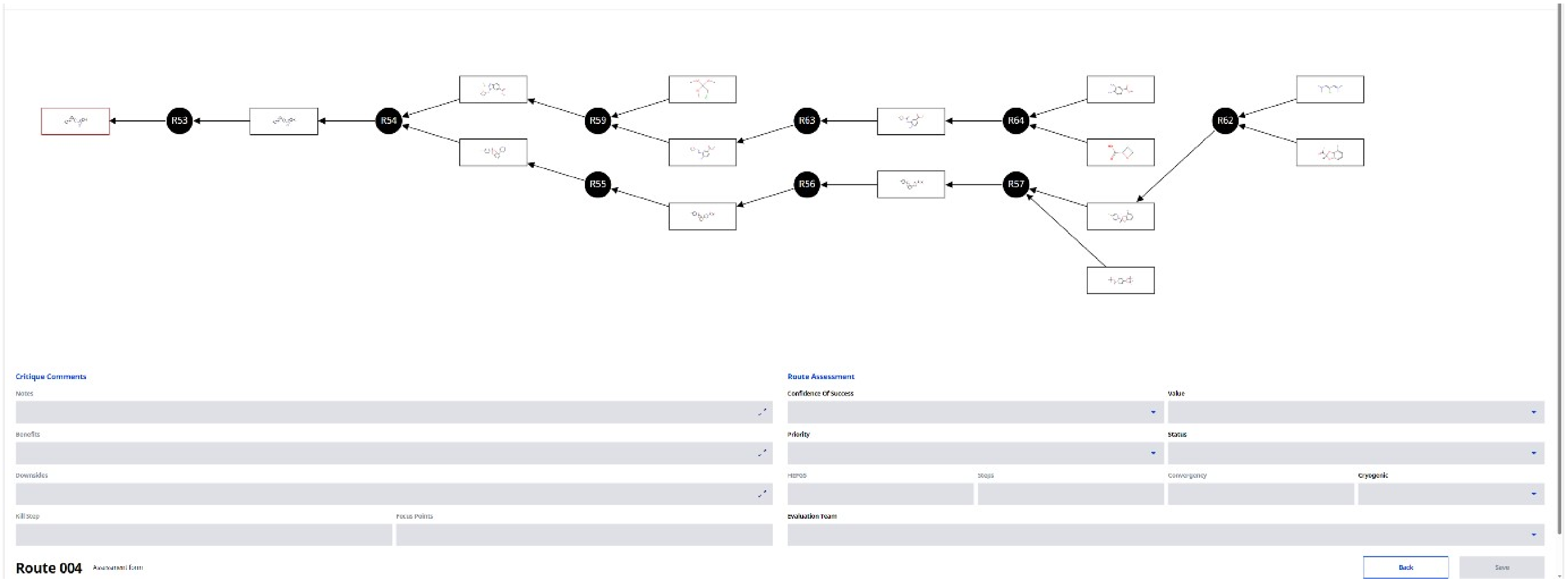
Steps

Convergency

Cryogenic

Evaluation Team

Route 003 Assessment form



Critique Comments

Notes

Benefits

Downsides

Kill Step

Focus Points

Route Assessment

Confidence Of Success

Value

Priority

Status

Heros

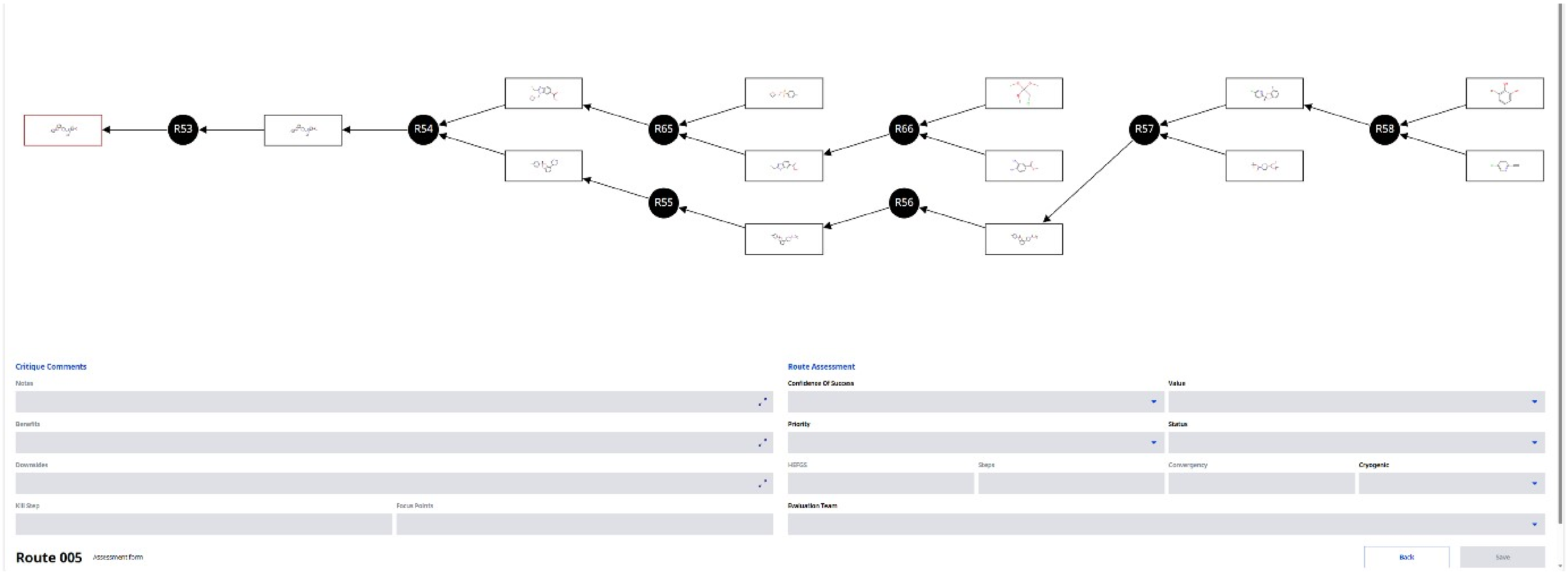
Steps

Convergency

Cryogenic

Evaluation Team

Route 004 Assessment form



Critique Comments

Notes

Benefits

Downsides

Kill Step

Focus Points

Route Assessment

Confidence Of Success

Value

Priority

Status

HEFGS

Steps

Convergency

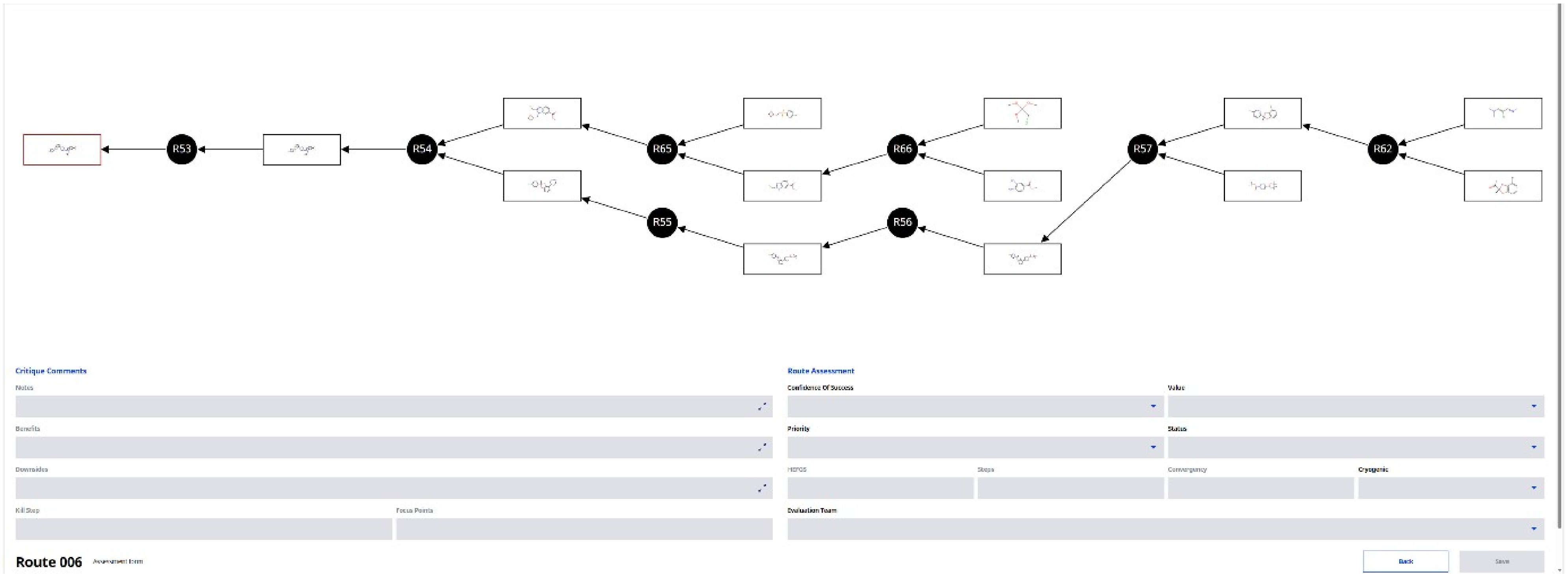
Cryogenic

Evaluation Team

Route 005 Assessment form

Back

Save



Critique Comments

Notes

Benefits

Downsides

Kill Step

Focus Points

Route Assessment

Confidence Of Success

Value

Priority

Status

HEFOS

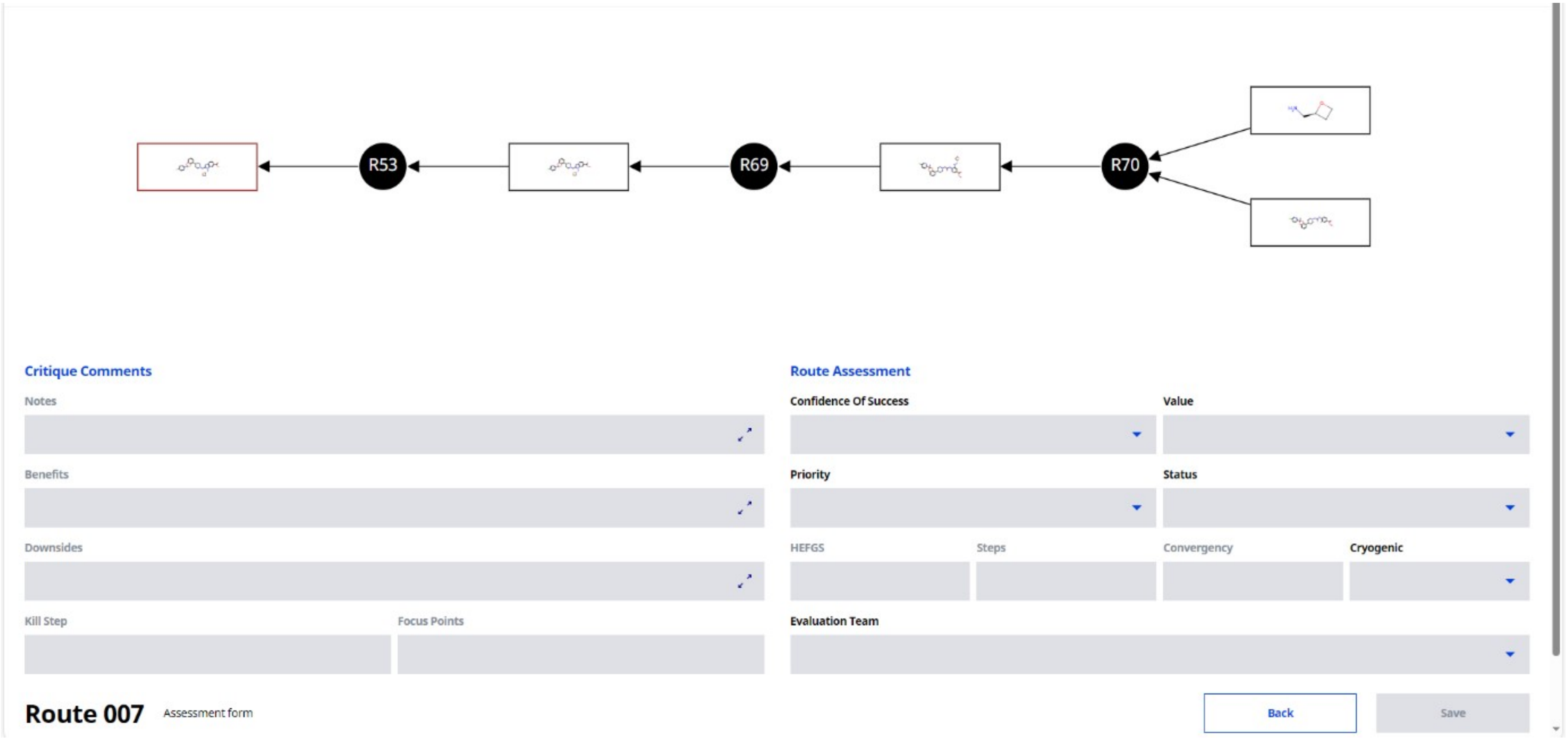
Steps

Convergency

Cryogenic

Evaluation Team

Route 006 Assessment form



Critique Comments

Notes

Benefits

Downsides

Kill Step	Focus Points

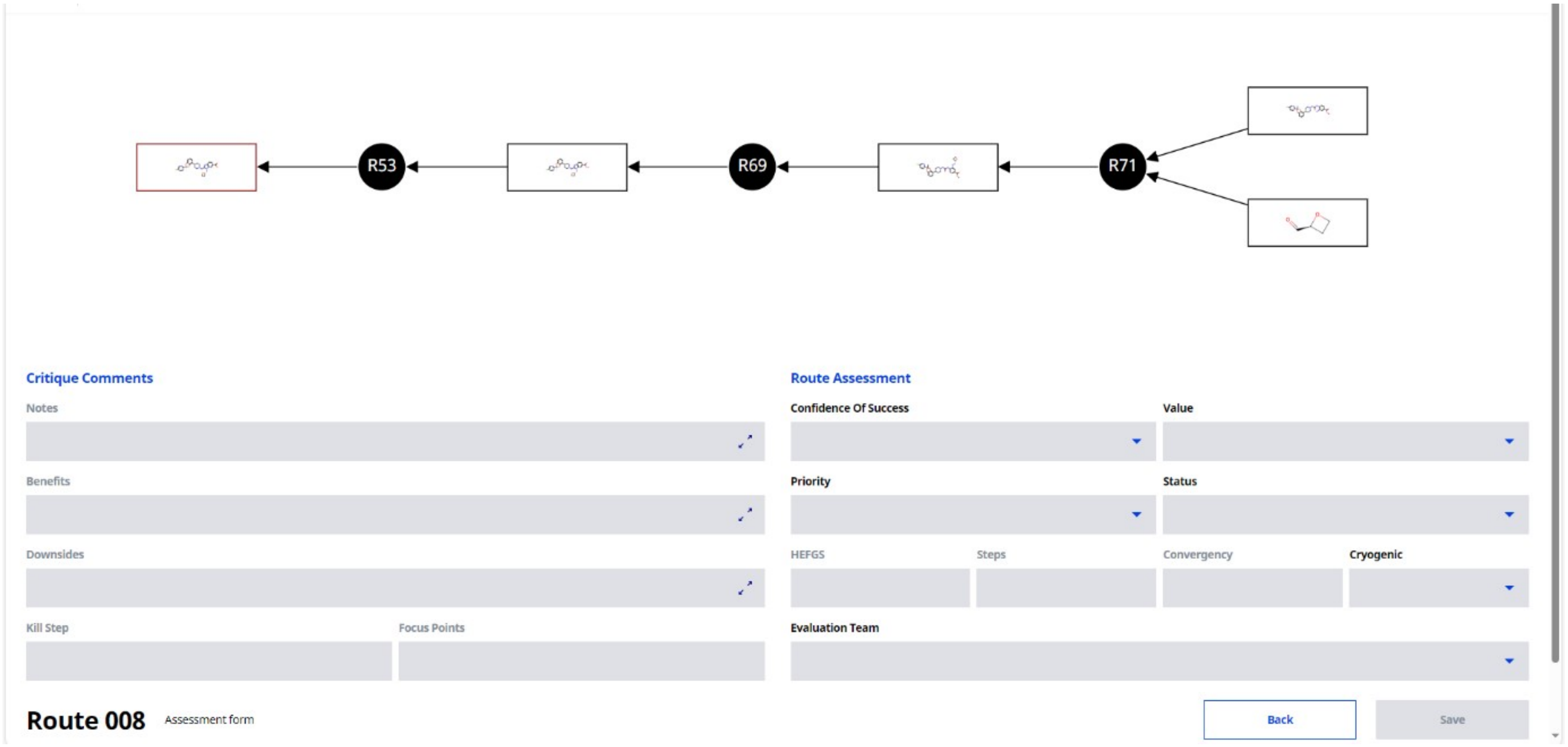
Route Assessment

Confidence Of Success	Value
Priority	Status

HEFGS	Steps	Convergency	Cryogenic

Evaluation Team

Route 007 Assessment form



Critique Comments

Notes

Benefits

Downsides

Kill Step

Focus Points

Route Assessment

Confidence Of Success

Value

Priority

Status

HEFGS

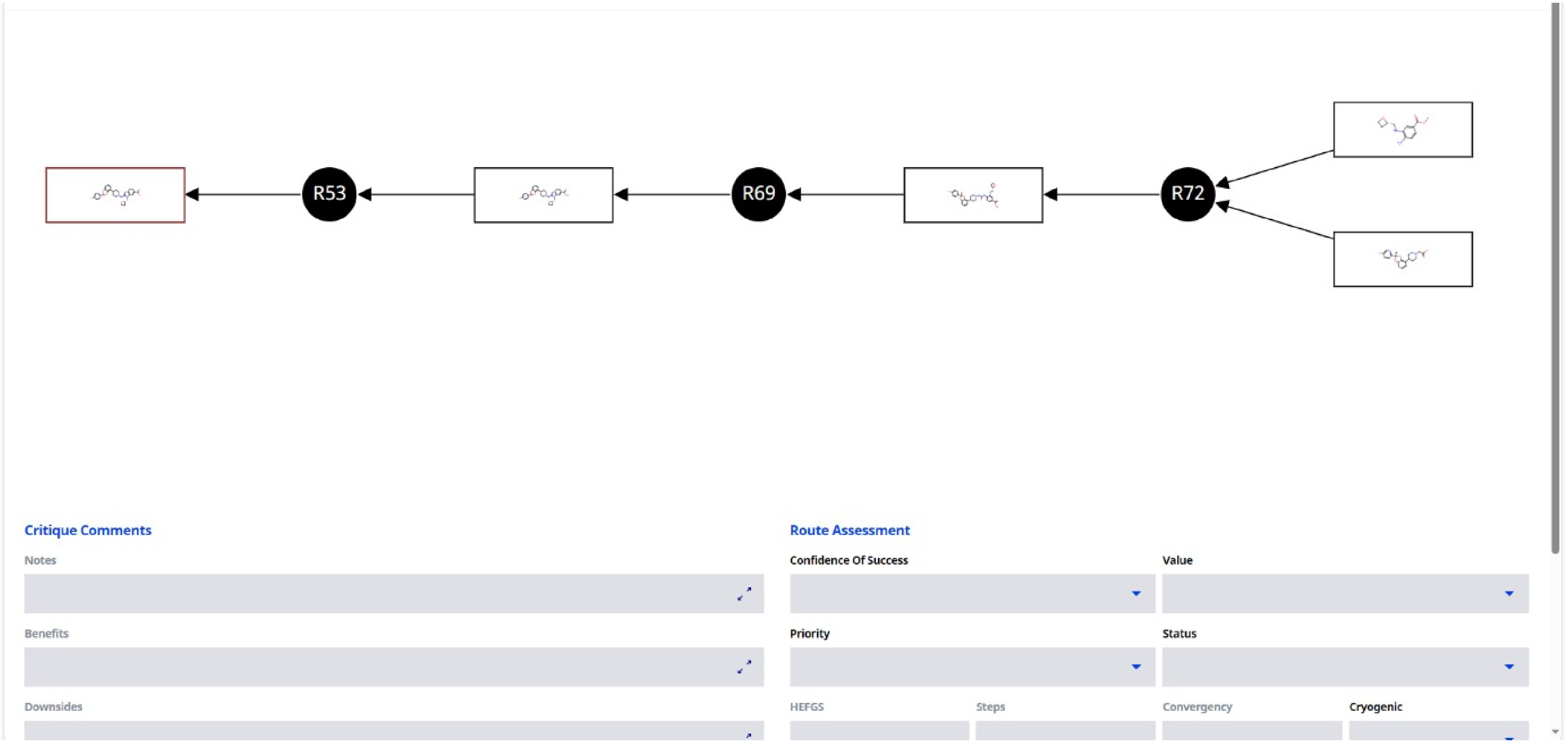
Steps

Convergency

Cryogenic

Evaluation Team

Route 008 Assessment form



Critique Comments

Notes

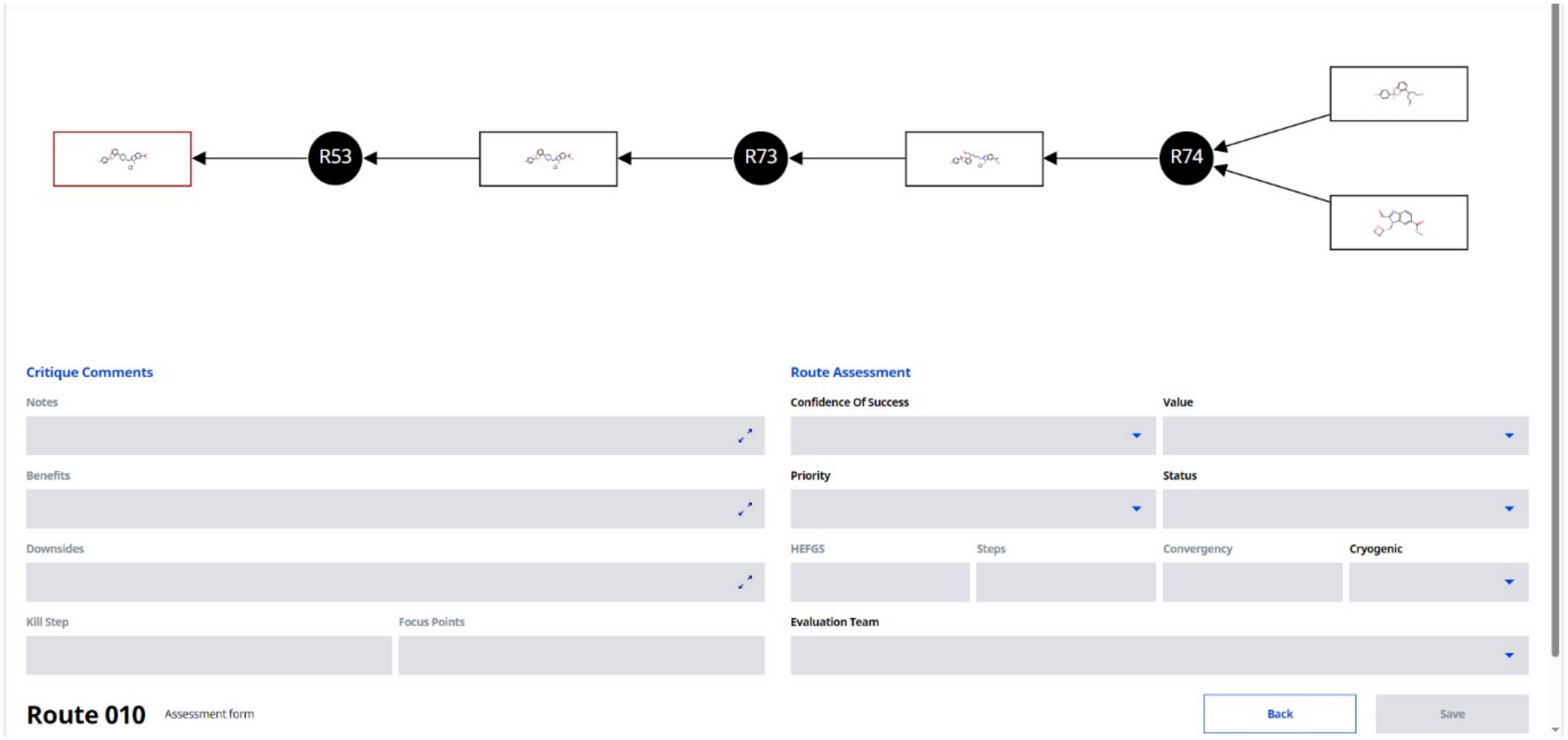
Benefits

Downsides

Route Assessment

Confidence Of Success	Value
Priority	Status

HEFGS	Steps	Convergency	Cryogenic



Critique Comments

Notes

Benefits

Downsides

Kill Step | Focus Points

Route Assessment

Confidence Of Success | Value

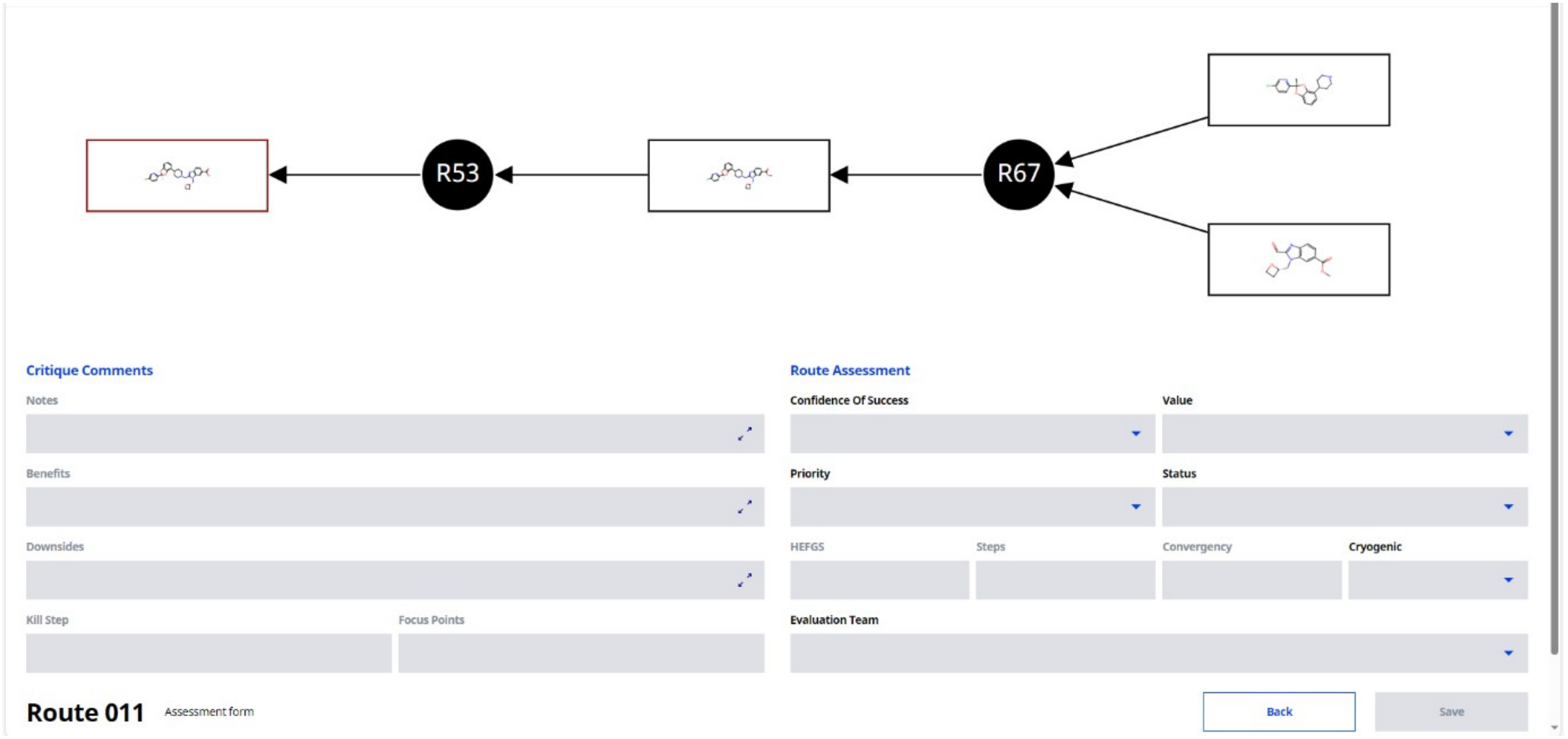
Priority | Status

HEFGS | Steps | Convergency | Cryogenic

Evaluation Team

Route 010 Assessment form

Back | Save



Critique Comments

Notes

Benefits

Downsides

Kill Step

Focus Points

Route Assessment

Confidence Of Success

Value

Priority

Status

HEFGS

Steps

Convergency

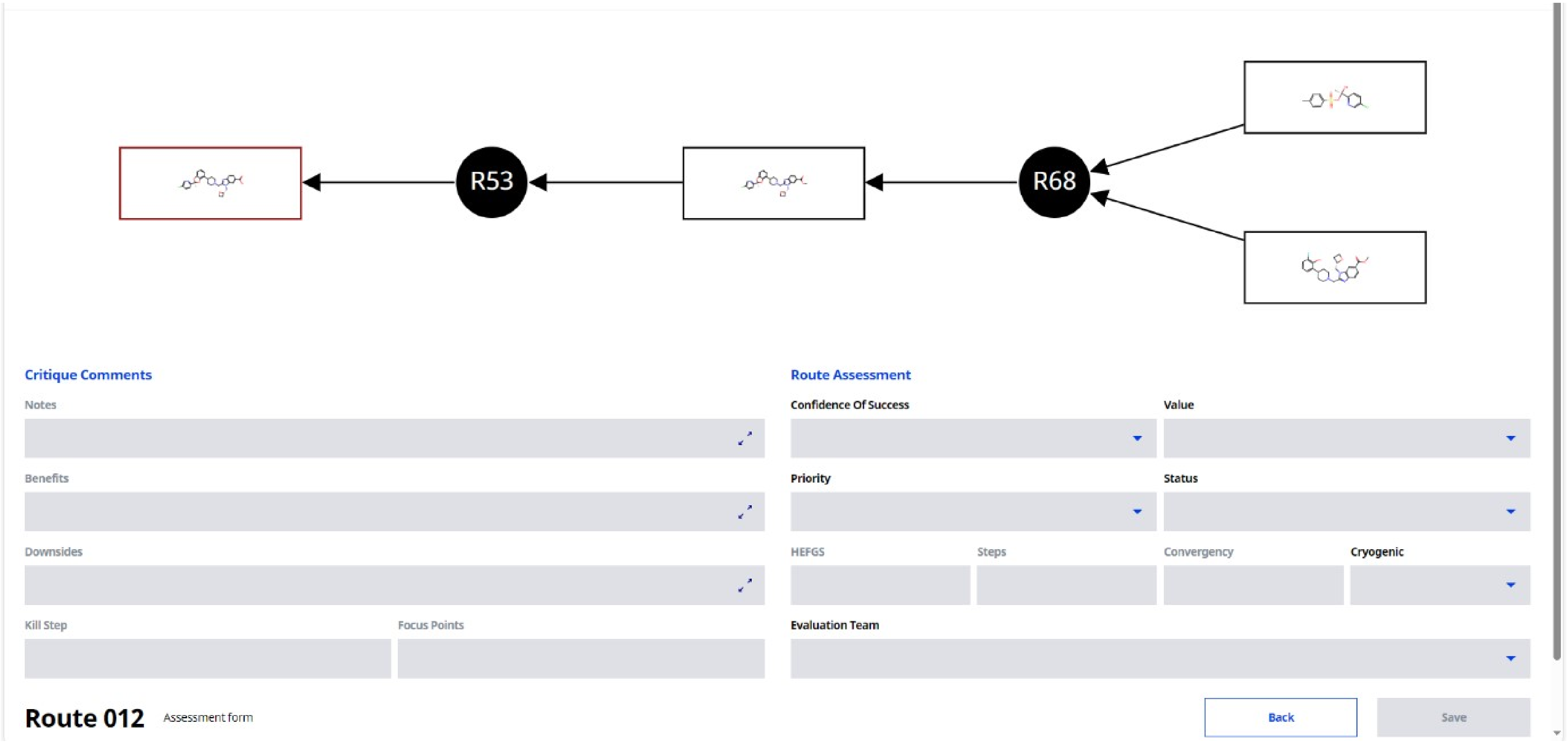
Cryogenic

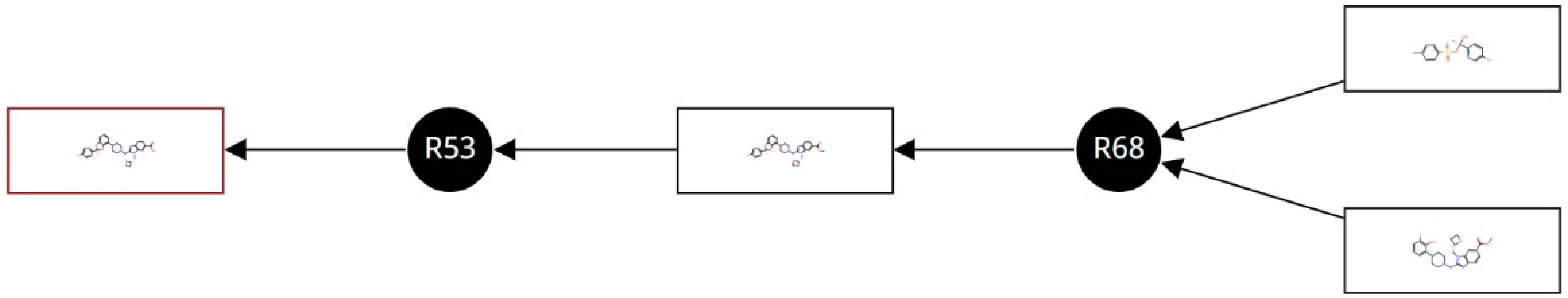
Evaluation Team

Route 011 Assessment form

Back

Save





Critique Comments

Notes	
Benefits	
Downsides	
Kill Step	
Focus Points	

Route Assessment

Confidence Of Success	
Value	
Priority	
Status	
HEFGS	
Steps	
Convergency	
Cryogenic	
Evaluation Team	

Route 012 Assessment form

ESI-3 – Idea suggestions obtained from predictive software (ASKCOS)

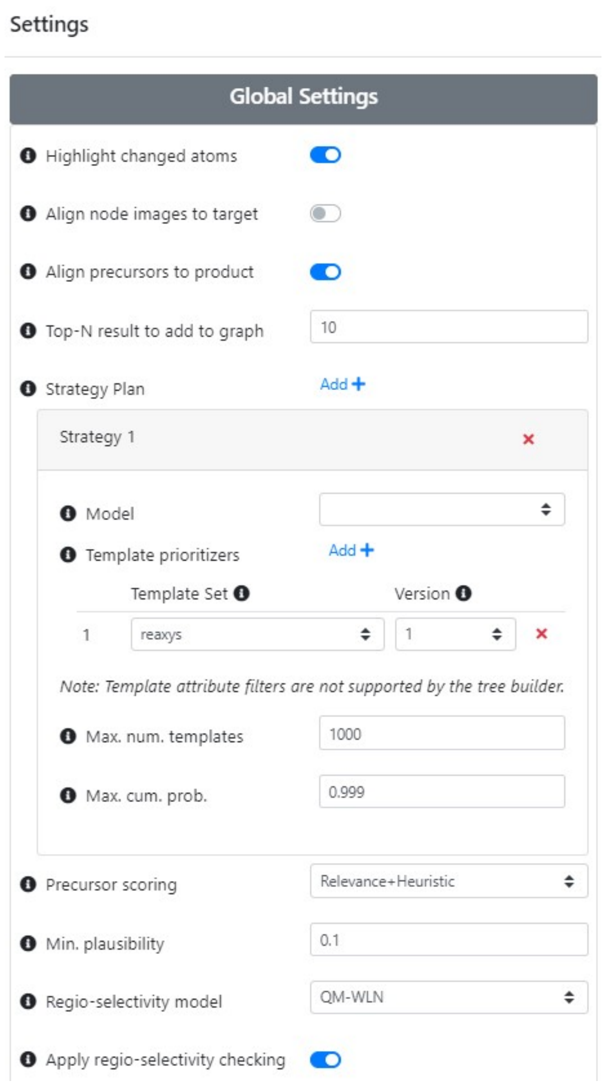
3.1 The target molecule (Lotiglipron) was introduced into ASKCOS software to obtain ideas synthetically generated. Details of the idea search are omitted but they can be found on the MIT website (open to the public)

3.2 ASKCOS model settings used.

ASKCOS stand for Automated System for Knowledge-Based Continuous Organic Synthesis. It is a software output created by the Machine Learning for Pharmaceutical Discovery and Synthesis Consortium (MLPDS), a collaboration between the pharmaceutical and biotechnology industries (including Pfizer) and the departments of Chemical Engineering, Chemistry, and Computer Science at the Massachusetts Institute of Technology (MIT, United States). Additional information available at https://mlpds.mit.edu/

- Predictions were generated using Pfizer-ASKCOS (private consortia member version)
- Public version generates similar outputs, and it is available at https://askcos.mit.edu/

The model settings used for a basic retrosynthesis search are displayed below:


Settings

Global Settings

Highlight changed atoms	On
Align node images to target	Off
Align precursors to product	On
Top-N result to add to graph	10
Strategy Plan	Add +

Strategy 1

Model	
Template prioritizers	Add +

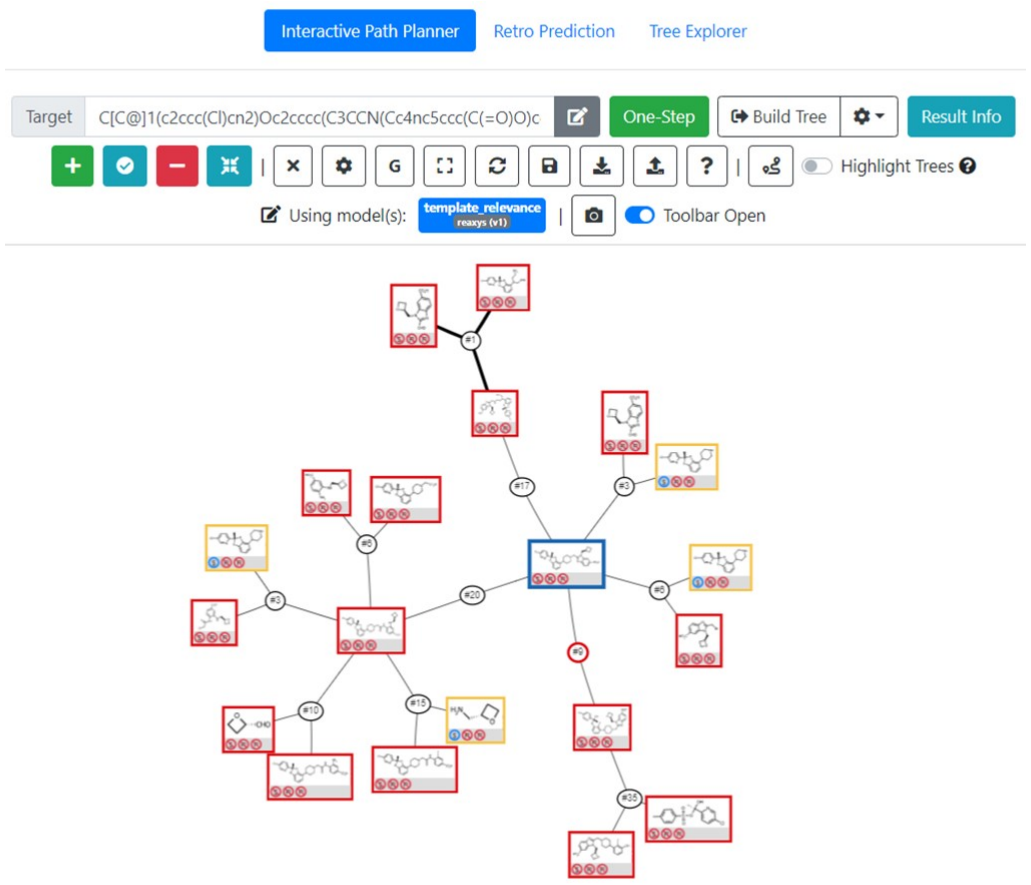
	Template Set	Version
1	reaxys	1

Note: Template attribute filters are not supported by the tree builder.

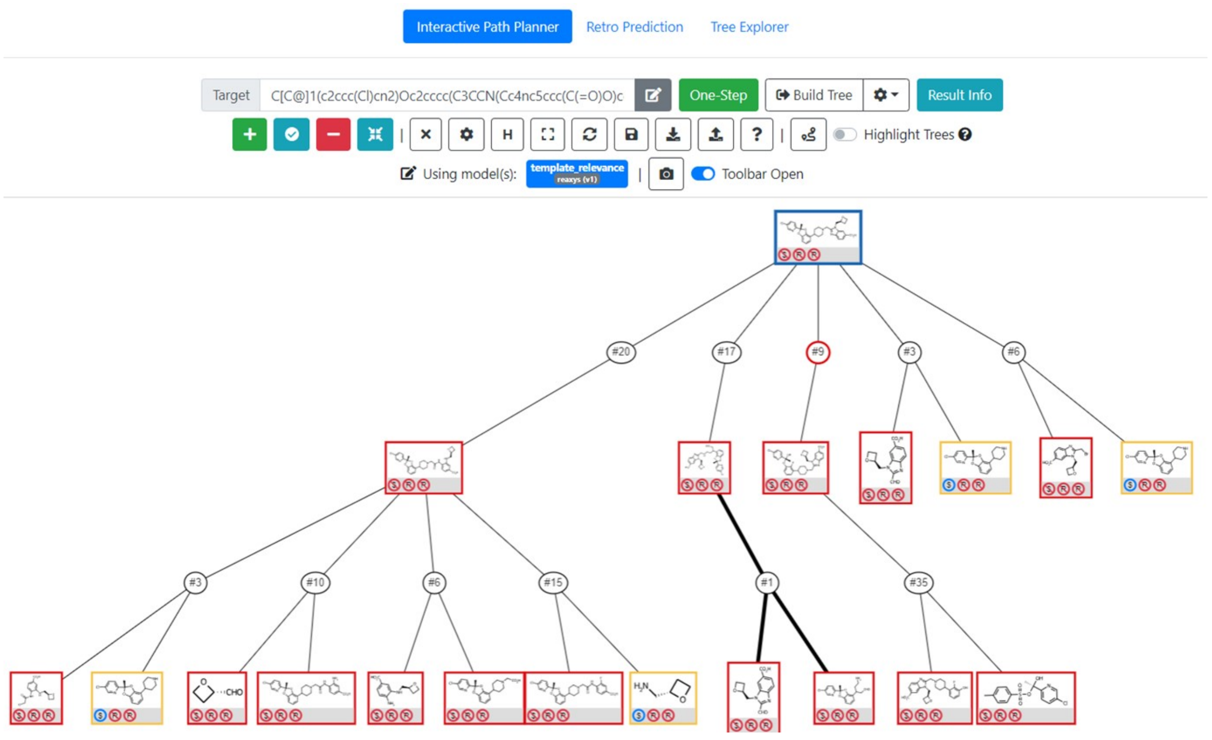
Max. num. templates	1000
Max. cum. prob.	0.999
Precursor scoring	Relevance+Heuristic
Min. plausibility	0.1
Regio-selectivity model	QM-WLN
Apply regio-selectivity checking	On

3.3 Filtered routes obtained by human aided prediction.

The output from a basic search is shown as follow. Network view:



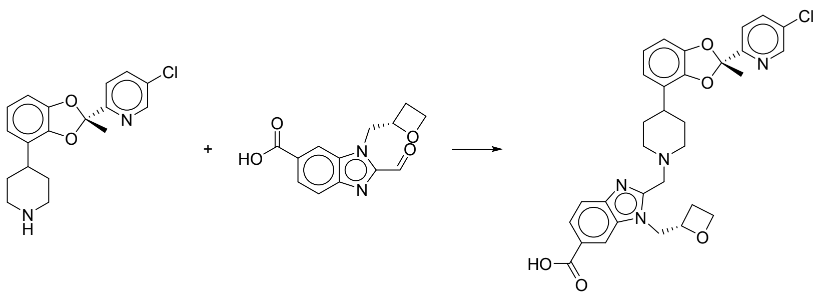
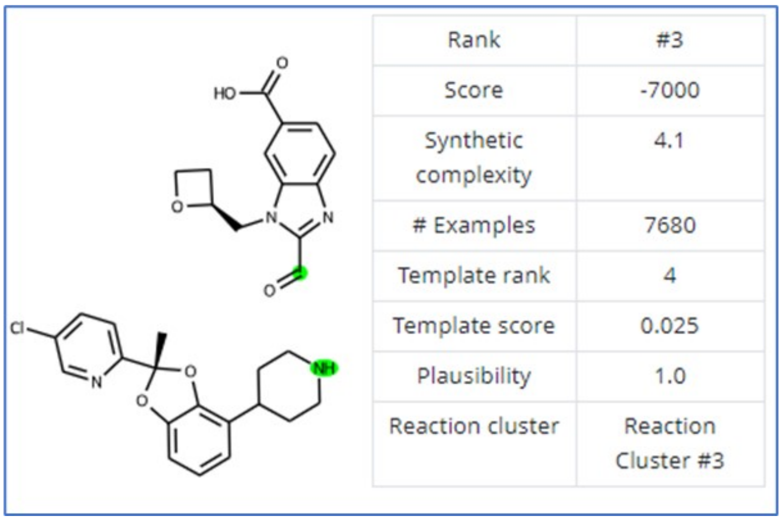
Alternative tree view:



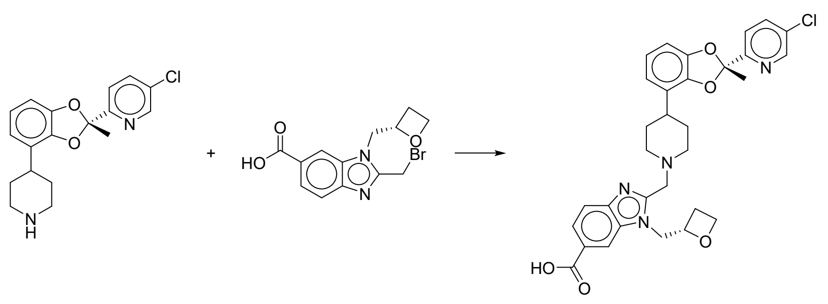
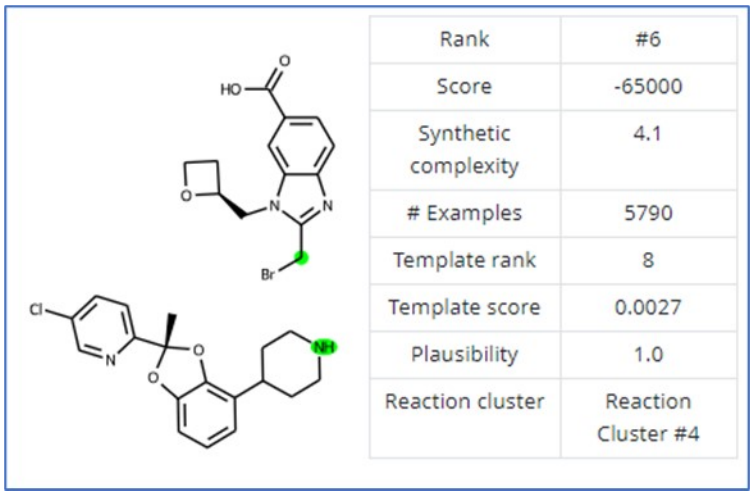
3.4 Description of the individual steps.

The disconnections shown in the trees above can be reproduced by using the images or reaction smiles provided in the tables below for each of the individual transformations. These details were use later to merge human and digitally generated ideas (Supplementary information 1 and 2).

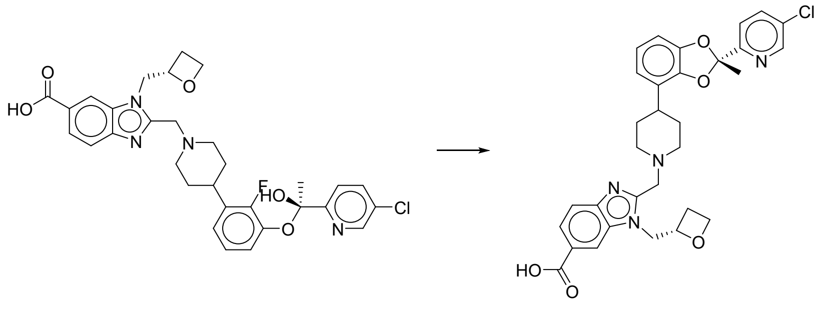
1st level disconnections from target

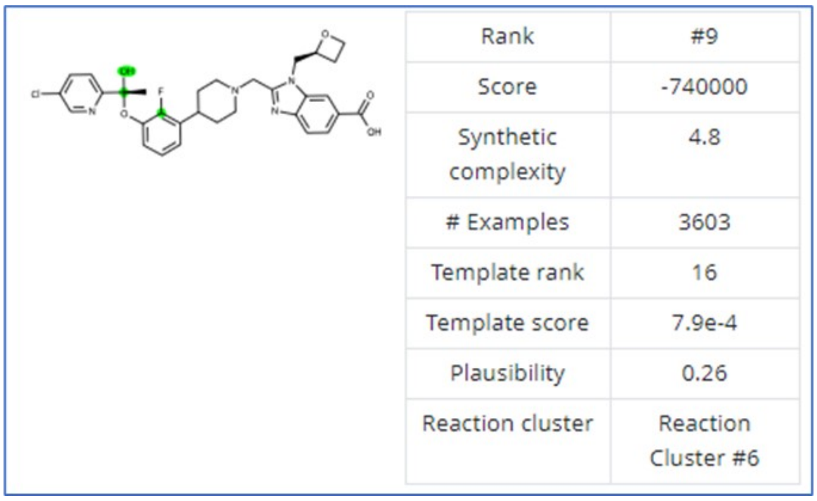
Reaction Node Number	3
Reaction	
Reaction SMILES	C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCNCC3)c2O1.O=Cc1nc2ccc(C(=O)O)cc2n1C[C@@H]1CCO1>>C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(Cc4nc5ccc(C(=O)O)cc5n4C[C@@H]4CCO4)CC3)c2O1
ASKCOS output	 Rank #3; Score -7000; Synthetic complexity 4.1; # Examples 7680; Template rank 4; Template score 0.025; Plausibility 1.0; Reaction cluster Reaction Cluster #3

Reaction Node Number	6

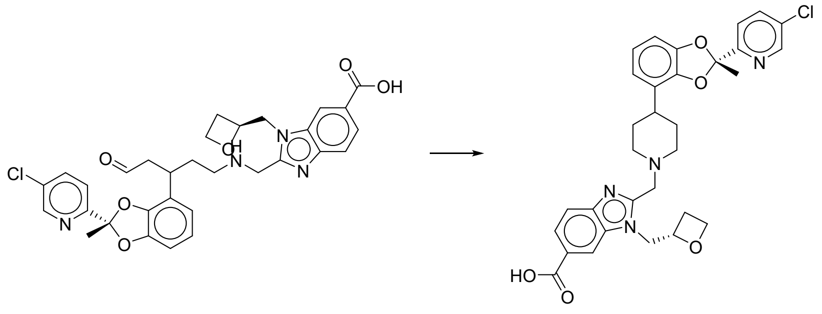
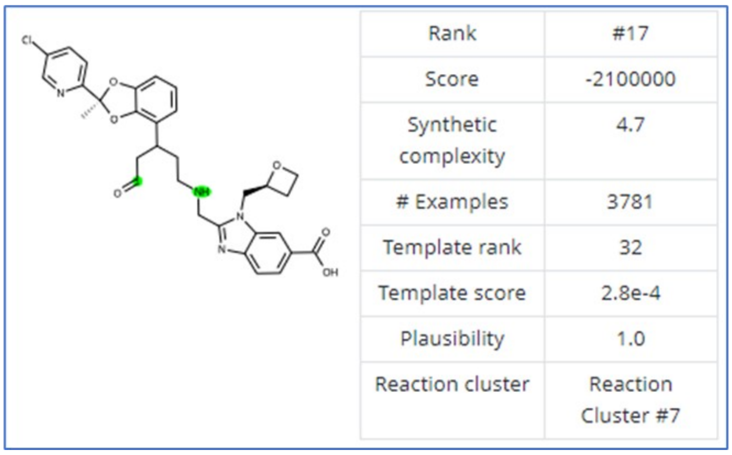
Reaction	
Reaction SMILES	C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCNCC3)c2O1.O=C(O)c1ccc2nc(CBr)n(C[C@@H]3CCO3)c2c1>>C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(Cc4nc5ccc(C(=O)O)cc5n4C[C@@H]4CCO4)CC3)c2O1
ASKCOS output	

Rank	#6
Score	-65000
Synthetic complexity	4.1
# Examples	5790
Template rank	8
Template score	0.0027
Plausibility	1.0
Reaction cluster	Reaction Cluster #4

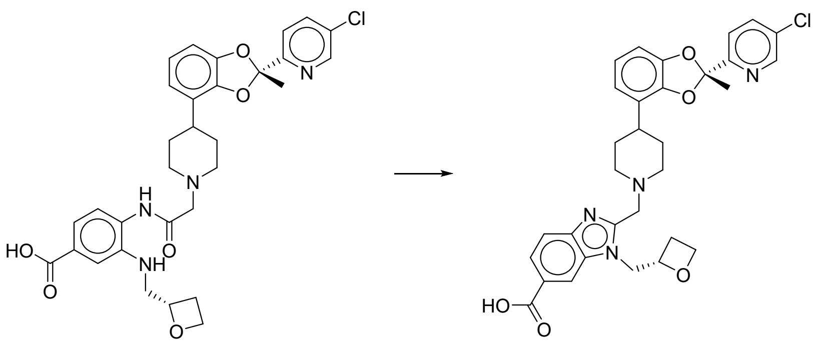
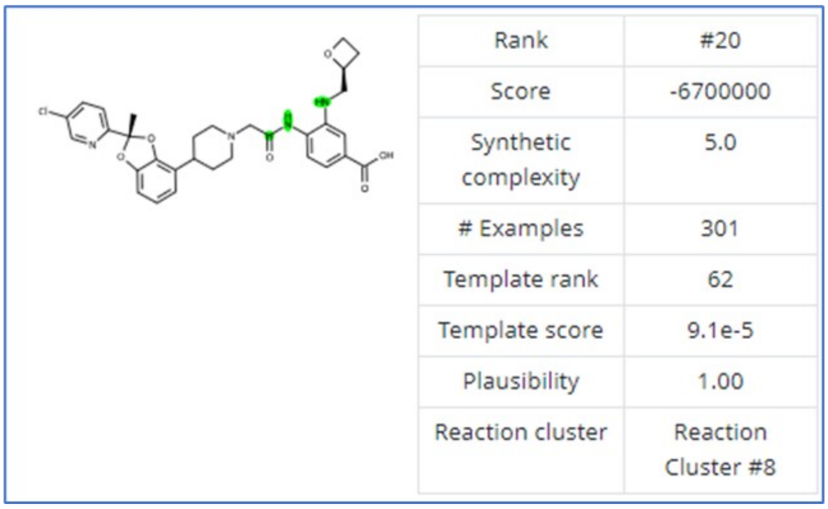
Reaction Node Number	9
Reaction	
Reaction SMILES	C[C@@](O)(Oc1cccc(C2CCN(Cc3nc4ccc(C(=O)O)cc4n3C[C@@H]3CCO3)CC2)c1F)c1ccc(Cl)cn1>>C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(Cc4nc5ccc(C(=O)O)cc5n4C[C@@H]4CCO4)CC3)c2O1

ASKCOS output	

Rank	#9
Score	-740000
Synthetic complexity	4.8
# Examples	3603
Template rank	16
Template score	7.9e-4
Plausibility	0.26
Reaction cluster	Reaction Cluster #6

Reaction Node Number	17
Reaction	
Reaction SMILES	C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C(CC=O)CCNCc3nc4ccc(C(=O)O)cc4n3C[C@@H]3CCO3)c2O1>>C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(Cc4nc5ccc(C(=O)O)cc5n4C[C@@H]4CCO4)CC3)c2O1
ASKCOS output	

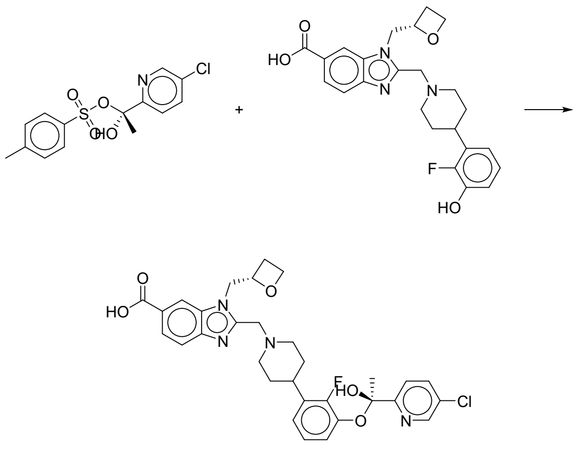
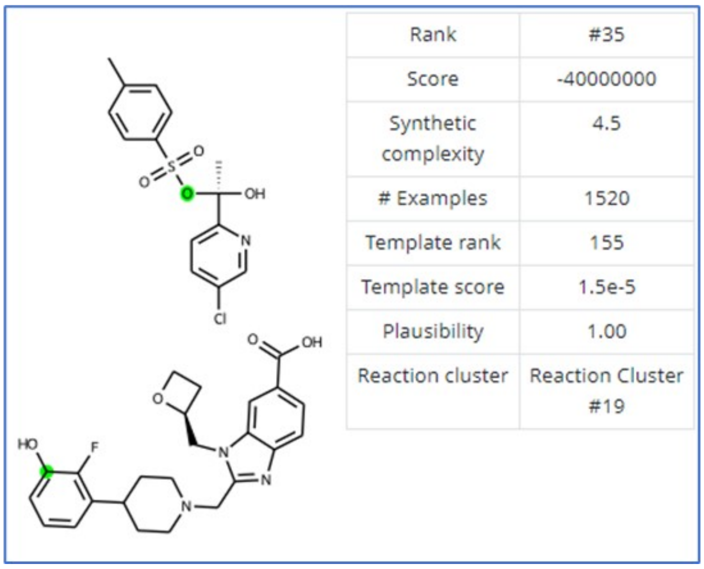
Rank	#17
Score	-2100000
Synthetic complexity	4.7
# Examples	3781
Template rank	32
Template score	2.8e-4
Plausibility	1.0
Reaction cluster	Reaction Cluster #7

Reaction Node Number	20
Reaction	 Cl O O N N H N O HO O NH O Cl O O N N N N O HO O
Reaction SMILES	C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(CC(=O)Nc4ccc(C(=O)O)cc4NC[C@@H]4CCO4)CC3)c2O1>>C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(Cc4nc5ccc(C(=O)O)cc5n4C[C@@H]4CCO4)CC3)c2O1
ASKCOS output	

Rank	#20
Score	-6700000
Synthetic complexity	5.0
# Examples	301
Template rank	62
Template score	9.1e-5
Plausibility	1.00
Reaction cluster	Reaction Cluster #8

1st branch disconnections (from reaction node 9)

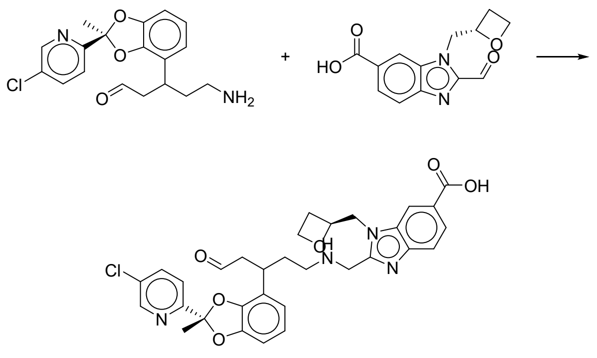
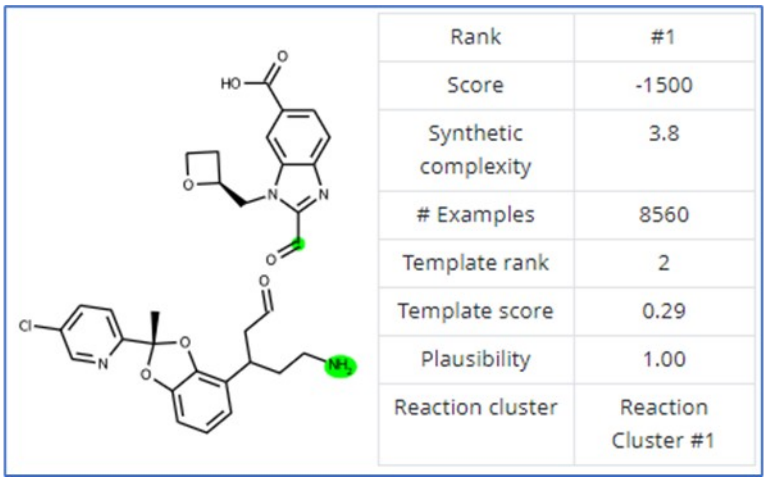
Reaction Node Number	35

Reaction	
Reaction SMILES	Cc1ccc(S(=O)(=O)O[C@@](C)(O)c2ccc(Cl)cn2)cc1.O=C(O)c1ccc2nc(CN3CCC(c4cccc(O)c4F)CC3)n(C[C@@H]3CCO3)c2c1>>C[C@@](O)(Oc1cccc(C2CCN(Cc3nc4ccc(C(=O)O)cc4n3C[C@@H]3CCO3)CC2)c1F)c1ccc(Cl)cn1
ASKCOS output	

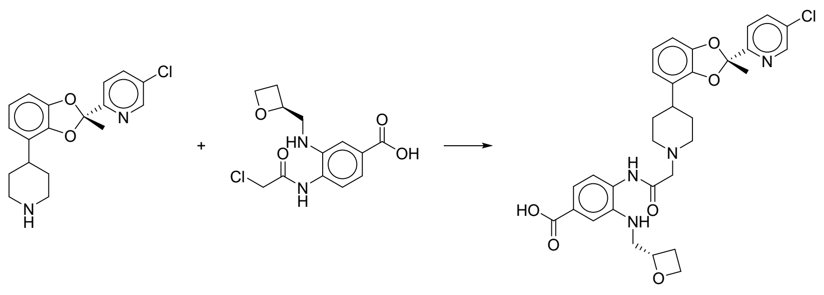
Rank	#35
Score	-40000000
Synthetic complexity	4.5
# Examples	1520
Template rank	155
Template score	1.5e-5
Plausibility	1.00
Reaction cluster	Reaction Cluster #19

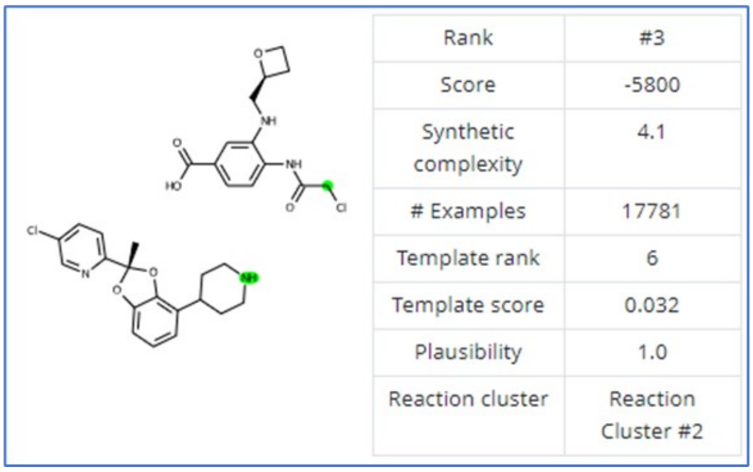
2nd branch disconnections (from reaction node 17)

Reaction Node Number	1

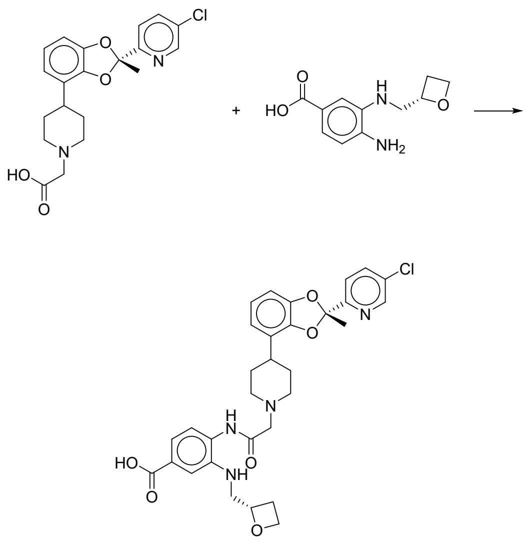
Reaction	
Reaction SMILES	C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C(CC=O)CCN)c2O1.O=Cc1nc2ccc(C(=O)O)cc2n1C[C@@H]1CCO1>>C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C(CC=O)CCNCc3nc4ccc(C(=O)O)cc4n3C[C@@H]3CCO3)c2O1
ASKCOS output	 Rank #1 Score -1500 Synthetic complexity 3.8 # Examples 8560 Template rank 2 Template score 0.29 Plausibility 1.00 Reaction cluster Reaction Cluster #1

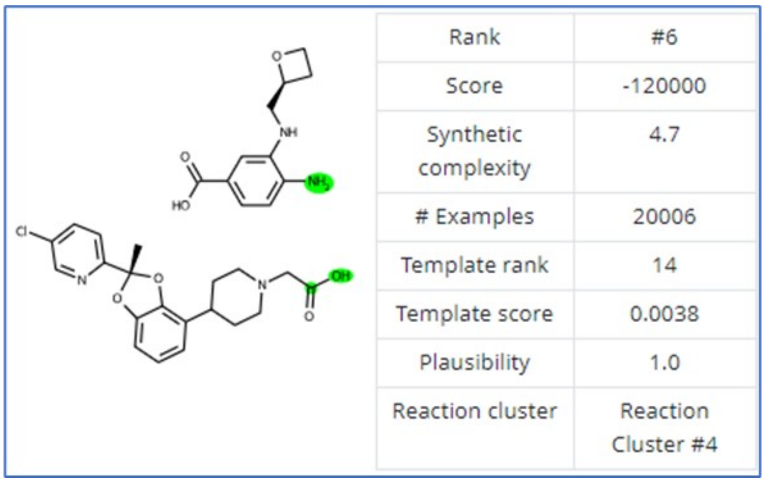
3rd branch disconnections (from reaction node 20)

Reaction Node Number	3
Reaction	
Reaction SMILES	C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCNCC3)c2O1.O=C(CCl)Nc1cc(C(=O)O)cc1NC[C@@H]1CCO1>>C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(CC(=O)Nc4ccc(C(=O)O)cc4NC[C@@H]4CCO4)CC3)c

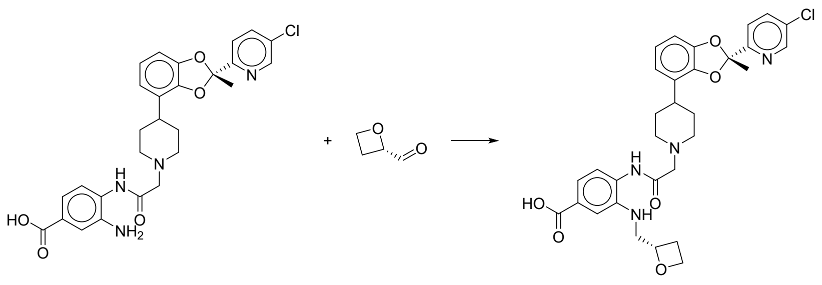
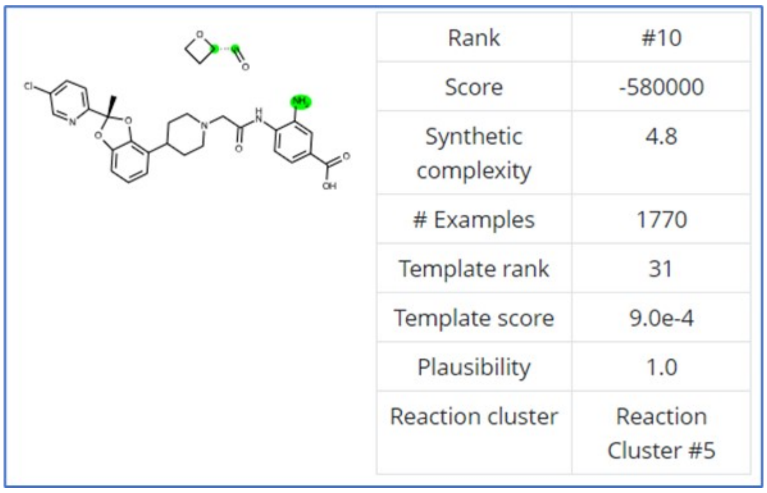
	2O1
ASKCOS output	

Rank	#3
Score	-5800
Synthetic complexity	4.1
# Examples	17781
Template rank	6
Template score	0.032
Plausibility	1.0
Reaction cluster	Reaction Cluster #2

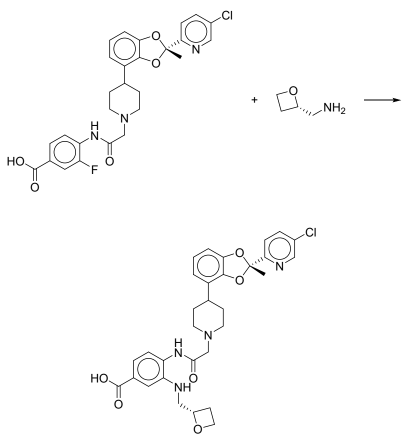
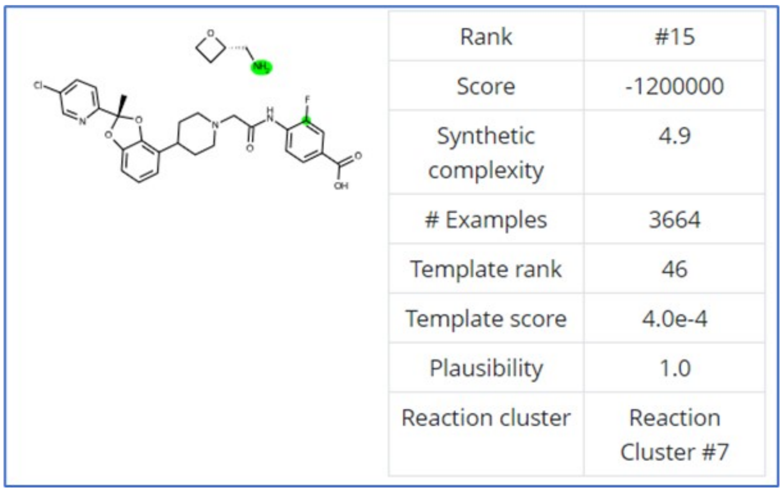
Reaction Node Number	6
Reaction	
Reaction SMILES	C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(CC(=O)O)CC3)c2O1.Nc1ccc(C(=O)O)cc1NC[C@@H]1CCO1>>C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(CC(=O)Nc4ccc(C(=O)O)cc4NC[C@@H]4CCO4)CC3)c2O1

ASKCOS output	

Rank	#6
Score	-120000
Synthetic complexity	4.7
# Examples	20006
Template rank	14
Template score	0.0038
Plausibility	1.0
Reaction cluster	Reaction Cluster #4

Reaction Node Number	10
Reaction	
Reaction SMILES	C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(CC(=O)Nc4ccc(C(=O)O)cc4N)CC3)c2O1.O=C[C@@H]1CCO1>>C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(CC(=O)Nc4ccc(C(=O)O)cc4NC[C@@H]4CCO4)CC3)c2O1
ASKCOS output	

Rank	#10
Score	-580000
Synthetic complexity	4.8
# Examples	1770
Template rank	31
Template score	9.0e-4
Plausibility	1.0
Reaction cluster	Reaction Cluster #5

Reaction Node Number	15
Reaction	
Reaction SMILES	C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(CC(=O)Nc4ccc(C(=O)O)cc4F)CC3)c2O1.NC[C@@H]1CCO1>>C[C@]1(c2ccc(Cl)cn2)Oc2cccc(C3CCN(CC(=O)Nc4ccc(C(=O)O)cc4NC[C@@H]4CCO4)CC3)c2O1
ASKCOS output	

Rank	#15
Score	-1200000
Synthetic complexity	4.9
# Examples	3664
Template rank	46
Template score	4.0e-4
Plausibility	1.0
Reaction cluster	Reaction Cluster #7

ESI-4 – Reproducing the graph database (using Neo4J) for Lotiglipron.

A set of queries is provided to reproduce the entire database presented in this paper.

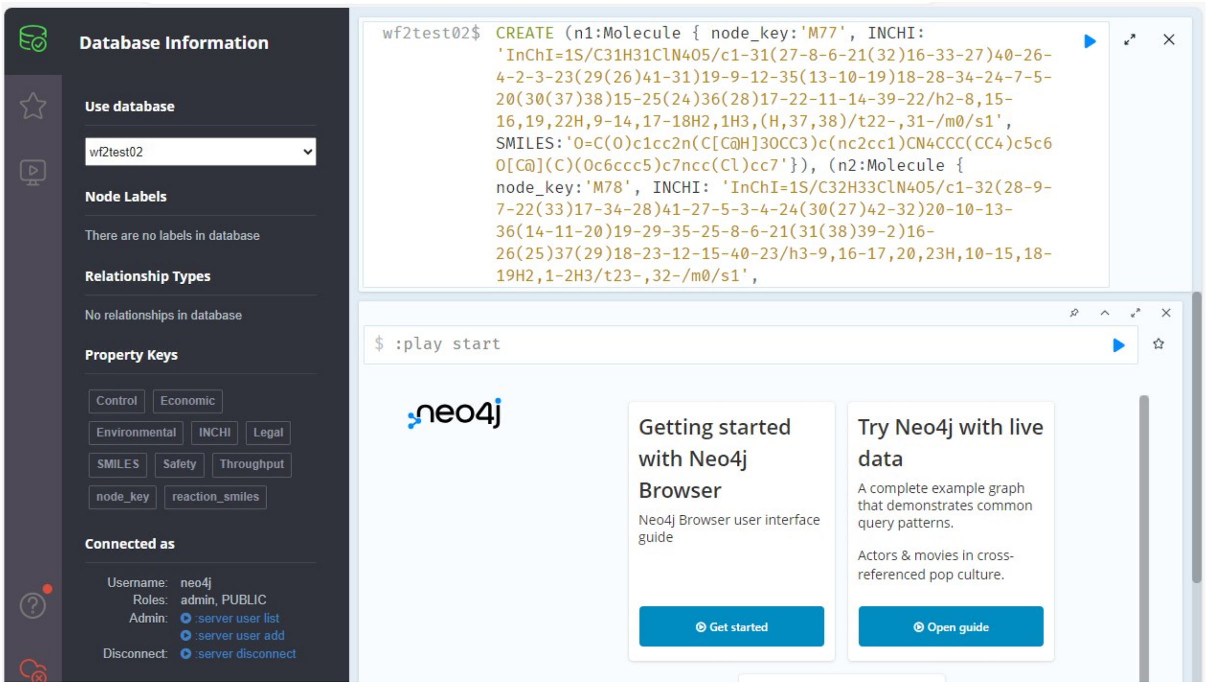
For this, a public or community version of neo4j can be used as the graph database (accessible from https://neo4j.com/). From this link, you can create a new account and an initial database. After this, the Lotiglipron network can be created following the instructions below:

The queries in Cypher (Neo4J native language) are provided each inside a box. Follow them step by step, copying and pasting the queries as one chunk, directly on the Neo4J browser. Then, to run the query press the blue arrow at the top right of the browser.

1. Query 1 to create all molecule nodes (copy and paste it into the neo4j browser):

```
CREATE (n1:Molecule { node_key:'M77', INCHI: 'InChI=1S/C31H31ClN4O5/c1-31(27-8-6-21(32)16-33-
27)40-26-4-2-3-23(29(26)41-31)19-9-12-35(13-10-19)18-28-34-24-7-5-20(30(37)38)15-25(24)36(28)17-22-
11-14-39-22/h2-8,15-16,19,22H,9-14,17-18H2,1H3,(H,37,38)/t22-,31-/m0/s1',
SMILES:'O=C(O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC4)c5c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7'
, Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n2:Molecule { node_key:'M78', INCHI: 'InChI=1S/C32H33ClN4O5/c1-32(28-9-7-22(33)17-34-28)41-27-5-
3-4-24(30(27)42-32)20-10-13-36(14-11-20)19-29-35-25-8-6-21(31(38)39-2)16-26(25)37(29)18-23-12-15-
40-23/h3-9,16-17,20,23H,10-15,18-19H2,1-2H3/t23-,32-/m0/s1',
SMILES:'COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC4)c5c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc
7', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n3:Molecule { node_key:'M79', INCHI: 'InChI=1S/C18H19ClN2O2/c1-18(16-6-5-13(19)11-21-16)22-15-4-
2-3-14(17(15)23-18)12-7-9-20-10-8-12/h2-6,11-12,20H,7-10H2,1H3/t18-/m0/s1',
SMILES:'N1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '}),
(n4:Molecule { node_key:'M80', INCHI: 'InChI=1S/C14H15ClN2O3/c1-19-14(18)9-2-3-11-12(6-9)17(13(7-
15)16-11)8-10-4-5-20-10/h2-3,6,10H,4-5,7-8H2,1H3/t10-/m0/s1',
SMILES:'COC(=O)c1cc2n(C[C@H]3OCC3)c(CCl)nc2cc1', Safety:' ', Environmental:' ', Legal:' ', Economic:'
', Control:' ', Throughput:' '}),  (n5:Molecule { node_key:'M81', INCHI:'InChI=1S/C23H27ClN2O4/c1-
22(2,3)30-21(27)26-12-10-15(11-13-26)17-6-5-7-18-20(17)29-23(4,28-18)19-9-8-16(24)14-25-19/h5-9,14-
15H,10-13H2,1-4H3/t23-/m0/s1',
SMILES:'CC(C)(C)OC(=O)N1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4', Safety:' ', Environmental:'
', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n6:Molecule { node_key:'M82', INCHI:'InChI=1S/C23H25ClN2O4/c1-22(2,3)30-21(27)26-12-10-15(11-13-
26)17-6-5-7-18-20(17)29-23(4,28-18)19-9-8-16(24)14-25-19/h5-10,14H,11-13H2,1-4H3/t23-/m0/s1',
SMILES:'CC(C)(C)OC(=O)N1CCC(=CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4', Safety:' ',
Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n7:Molecule { node_key:'M83', INCHI:'InChI=1S/C13H9BrClNO2/c1-13(11-6-5-8(15)7-16-11)17-10-4-2-
3-9(14)12(10)18-13/h2-7H,1H3/t13-/m0/s1', SMILES:'Clc1cnc(cc1)[C@]2(C)Oc3c(O2)cccc3Br', Safety:' ',
Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n8:Molecule { node_key:'M84', INCHI:'InChI=1S/C16H28BNO4/c1-14(2,3)20-13(19)18-10-8-12(9-11-
18)17-21-15(4,5)16(6,7)22-17/h8H,9-11H2,1-7H3',
SMILES:'CC(C)(C)OC(=O)N1CCC(=CC1)B2OC(C)(C)C(C)(O2)C', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '}),  (n9:Molecule { node_key:'M85', INCHI:'InChI=1S/C7H4ClN/c1-
2-7-4-3-6(8)5-9-7/h1,3-5H', SMILES:'C#Cc1ncc(Cl)cc1', Safety:' ', Environmental:' ', Legal:' ', Economic:' ',
Control:' ', Throughput:' '}),  (n10:Molecule { node_key:'M86', INCHI:'InChI=1S/C6H5BrO2/c7-4-2-1-3-
5(8)6(4)9/h1-3,8-9H', SMILES:'c1cc(Br)c(O)c(c1)O', Safety:' ', Environmental:' ', Legal:' ', Economic:' ',
Control:' ', Throughput:' '}),  (n11:Molecule { node_key:'M87', INCHI:'InChI=1S/C12H16N2O3/c1-16-
12(15)8-2-3-10(13)11(6-8)14-7-9-4-5-17-9/h2-3,6,9,14H,4-5,7,13H2,1H3/t9-/m0/s1',
SMILES:'C1O[C@@H](C1)CNc2c(N)ccc(c2)C(=O)OC', Safety:' ', Environmental:' ', Legal:' ', Economic:' ',
Control:' ', Throughput:' '}),  (n12:Molecule { node_key:'M88', INCHI:'InChI=1S/C5H11ClO3/c1-7-5(4-6,8-
2)9-3/h4H2,1-3H3', SMILES:'COC(OC)(OC)CCl', Safety:' ', Environmental:' ', Legal:' ', Economic:' ',
Control:' ', Throughput:' '}), (n13:Molecule { node_key:'M89', INCHI:'InChI=1S/C12H14N2O5/c1-18-
12(15)8-2-3-11(14(16)17)10(6-8)13-7-9-4-5-19-9/h2-3,6,9,13H,4-5,7H2,1H3/t9-/m0/s1',
SMILES:'C1O[C@@H](C1)CNc2c(N(=O)=O)ccc(c2)C(=O)OC', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '}), (n14:Molecule { node_key:'M90', INCHI:'InChI=1S/C4H9NO/c5-
3-4-1-2-6-4/h4H,1-3,5H2/t4-/m0/s1', SMILES:'NC[C@H]1OCC1', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '}),
(n15:Molecule { node_key:'M91', INCHI:'InChI=1S/C8H6FNO4/c1-14-8(11)5-2-3-7(10(12)13)6(9)4-5/h2-
4H,1H3', SMILES:'[O-][N+](=O)c1c(F)cc(cc1)C(=O)OC', Safety:' ', Environmental:' ', Legal:' ', Economic:' ',
Control:' ', Throughput:' '}), (n16:Molecule { node_key:'M92', INCHI:'InChI=1S/C7H14ClN2/c1-9(2)5-
7(8)6-10(3)4/h5-6H,1-4H3/q+1', SMILES:'CN(C)/C=C(Cl)/C=[N+](C)C', Safety:' ', Environmental:' ', Legal:'
', Economic:' ', Control:' ', Throughput:' '}),
(n17:Molecule { node_key:'M93', INCHI:'InChI=1S/C10H9BrO3/c1-6(12)10(2)13-8-5-3-4-7(11)9(8)14-
10/h3-5H,1-2H3/t10-/m0/s1', SMILES:'c1cc2O[C@](C)(C(=O)C)Oc2c(c1)Br', Safety:' ', Environmental:' ',
Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n18:Molecule { node_key:'M94', INCHI:'InChI=1S/C12H14N2O4/c1-17-12(16)7-2-3-8(13)9(6-7)14-
11(15)10-4-5-18-10/h2-3,6,10H,4-5,13H2,1H3,(H,14,15)/t10-/m0/s1',
SMILES:'C1O[C@@H](C1)C(=O)Nc2c(N)ccc(c2)C(=O)OC', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '}), (n19:Molecule { node_key:'M95', INCHI:'InChI=1S/C4H6O3/c5-
4(6)3-1-2-7-3/h3H,1-2H2,(H,5,6)/t3-/m0/s1', SMILES:'C1O[C@@H](C1)C(O)=O', Safety:' ',
Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n20:Molecule { node_key:'M96', INCHI:'InChI=1S/C8H10N2O2/c1-12-8(11)5-2-3-6(9)7(10)4-5/h2-4H,9-
10H2,1H3', SMILES:'Nc1c(N)cc(cc1)C(=O)OC', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:'
', Throughput:' '}), (n21:Molecule { node_key:'M97', INCHI:'InChI=1S/C10H9ClN2O2/c1-15-10(14)6-2-3-
7-8(4-6)13-9(5-11)12-7/h2-4H,5H2,1H3,(H,12,13)', SMILES:'COC(=O)c1cc2c(cc1)nc([nH]2)CCl', Safety:' ',
Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n22:Molecule { node_key:'M98', INCHI:'InChI=1S/C11H14O4S/c1-9-2-4-11(5-3-9)16(12,13)15-8-10-6-7-
14-10/h2-5,10H,6-8H2,1H3/t10-/m0/s1', SMILES:'Cc1ccc(cc1)S(=O)(=O)OC[C@H]2OCC2', Safety:' ',
Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n23:Molecule { node_key:'M99', INCHI:'InChI=1S/C14H14N2O4/c1-19-14(18)9-2-3-11-12(6-9)16(13(8-
17)15-11)7-10-4-5-20-10/h2-3,6,8,10H,4-5,7H2,1H3/t10-/m0/s1',
SMILES:'COC(=O)c1cc2n(C[C@H]3OCC3)c(C=O)nc2cc1', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '}),
(n24:Molecule { node_key:'M100', INCHI:'InChI=1S/C25H28FN3O4/c1-32-25(31)17-5-6-21-22(13-
17)29(14-18-9-12-33-18)23(27-21)15-28-10-7-16(8-11-28)19-3-2-4-20(26)24(19)30/h2-6,13,16,18,30H,7-
12,14-15H2,1H3/t18-/m0/s1',
SMILES:'Fc1c(O)c(ccc1)C2CCN(CC2)Cc3n(C[C@H]4OCC4)c5c(n3)ccc(c5)C(=O)OC', Safety:' ',
Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n25:Molecule { node_key:'M101', INCHI:'InChI=1S/C14H14ClNO4S/c1-10-3-6-12(7-4-10)21(18,19)20-
14(2,17)13-8-5-11(15)9-16-13/h3-9,17H,1-2H3/t14-/m1/s1',
SMILES:'Cc1ccc(cc1)S(=O)(=O)O[C@@](C)(O)c2ncc(Cl)cc2', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '}), (n26:Molecule { node_key:'M102', INCHI:'
InChI=1S/C32H35ClN4O6/c1-32(28-9-7-22(33)17-35-28)42-27-5-3-4-24(30(27)43-32)20-10-13-37(14-11-
20)19-29(38)36-25-8-6-21(31(39)40-2)16-26(25)34-18-23-12-15-41-23/h3-9,16-17,20,23,34H,10-15,18-
19H2,1-2H3,(H,36,38)/t23-,32-/m0/s1',
SMILES:'C1O[C@@H](C1)CNc2c(ccc(c2)C(OC)=O)NC(=O)CN3CCC(CC3)c4c5O[C@](C)(Oc5ccc4)c6n
cc(Cl)cc6', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n27:Molecule { node_key:'M103', INCHI:'InChI=1S/C28H27ClFN3O5/c1-28(24-9-7-19(29)15-31-24)37-
23-5-3-4-20(26(23)38-28)17-10-12-33(13-11-17)16-25(34)32-22-8-6-18(14-21(22)30)27(35)36-2/h3-9,14-
15,17H,10-13,16H2,1-2H3,(H,32,34)/t28-/m0/s1',
SMILES:'COC(=O)c1cc(F)c(cc1)NC(=O)CN2CCC(CC2)c3c4O[C@](C)(Oc4ccc3)c5ncc(Cl)cc5', Safety:' ',
Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n28:Molecule { node_key:'M104', INCHI:'InChI=1S/C4H6O2/c5-3-4-1-2-6-4/h3-4H,1-2H2/t4-/m0/s1',
SMILES:'O=C[C@H]1OCC1', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:'
'}), (n29:Molecule { node_key:'M105', INCHI:'InChI=1S/C28H29ClN4O5/c1-28(24-9-7-19(29)15-31-
24)37-23-5-3-4-20(26(23)38-28)17-10-12-33(13-11-17)16-25(34)32-22-8-6-18(14-21(22)30)27(35)36-2/h3-
9,14-15,17H,10-13,16,30H2,1-2H3,(H,32,34)/t28-/m0/s1',
SMILES:'COC(=O)c1cc(N)c(cc1)NC(=O)CN2CCC(CC2)c3c4O[C@](C)(Oc4ccc3)c5ncc(Cl)cc5', Safety:' ',
Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n30:Molecule { node_key:'M106', INCHI:' InChI=1S/C20H21ClN2O4/c1-20(17-6-5-14(21)11-22-17)26-16-
4-2-3-15(19(16)27-20)13-7-9-23(10-8-13)12-18(24)25/h2-6,11,13H,7-10,12H2,1H3,(H,24,25)/t20-/m0/s1',
SMILES:'O=C(O)CN1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4', Safety:' ', Environmental:' ',
Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n31:Molecule { node_key:'M107', INCHI:'InChI=1S/C32H33ClN4O6/c1-32(28-9-7-22(33)17-35-28)42-27-
5-3-4-24(30(27)43-32)20(11-14-38)10-13-34-18-29-36-25-8-6-21(31(39)40-2)16-26(25)37(29)19-23-12-15-
41-23/h3-9,14,16-17,20,23,34H,10-13,15,18-19H2,1-2H3/t20?,23-,32-/m0/s1',
SMILES:'COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CNCCC(CC=O)c4c5O[C@](C)(Oc5ccc4)c6ncc(Cl)c
c6', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n32:Molecule { node_key:'M108', INCHI:'InChI=1S/C18H19ClN2O3/c1-18(16-6-5-13(19)11-21-16)23-15-
4-2-3-14(17(15)24-18)12(7-9-20)8-10-22/h2-6,10-12H,7-9,20H2,1H3/t12?,18-/m0/s1',
SMILES:'NCCC(CC=O)c1c2O[C@](C)(Oc2ccc1)c3ncc(Cl)cc3', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '})
```

Copying this query in neo4j browser will look like this:



```
wf2test02$ CREATE (n1:Molecule { node_key:'M77', INCHI:
           'InChI=1S/C31H31ClN4O5/c1-31(27-8-6-21(32)16-33-27)40-26-
           4-2-3-23(29(26)41-31)19-9-12-35(13-10-19)18-28-34-24-7-5-
           20(30(37)38)15-25(24)36(28)17-22-11-14-39-22/h2-8,15-
           16,19,22H,9-14,17-18H2,1H3,(H,37,38)/t22-,31-/m0/s1',
           SMILES:'O=C(O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC4)c5c6
           O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7'}), (n2:Molecule {
           node_key:'M78', INCHI: 'InChI=1S/C32H33ClN4O5/c1-32(28-9-
           7-22(33)17-34-28)41-27-5-3-4-24(30(27)42-32)20-10-13-
           36(14-11-20)19-29-35-25-8-6-21(31(38)39-2)16-
           26(25)37(29)18-23-12-15-40-23/h3-9,16-17,20,23H,10-15,18-
           19H2,1-2H3/t23-,32-/m0/s1',
```

```
$ :play start
```

Database Information

Use database

wf2test02

Node Labels

There are no labels in database

Relationship Types

No relationships in database

Property Keys

Control Economic Environmental INCHI Legal SMILES Safety Throughput node_key reaction_smiles

Connected as

Username: neo4j
Roles: admin, PUBLIC
Admin: :server user list
:server user add
Disconnect: :server disconnect

Getting started with Neo4j Browser

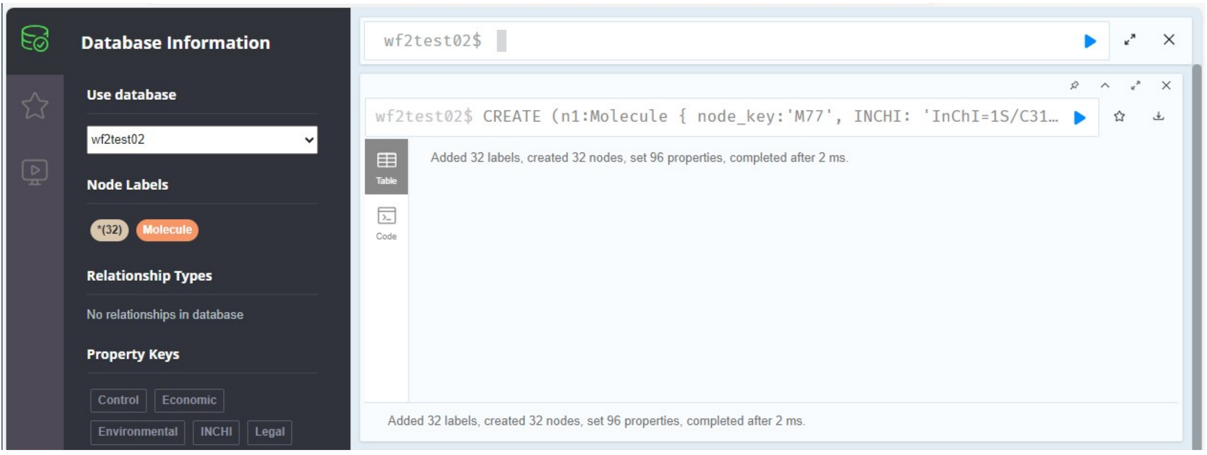
Neo4j Browser user interface guide

Try Neo4j with live data

A complete example graph that demonstrates common query patterns.

Actors & movies in cross-referenced pop culture.

Pressing the blue arrow at the top will run the query with the following answer:



```
wf2test02$
```

```
wf2test02$ CREATE (n1:Molecule { node_key:'M77', INCHI: 'InChI=1S/C31…
```

```
Added 32 labels, created 32 nodes, set 96 properties, completed after 2 ms.
```

Database Information

Use database

wf2test02

Node Labels

*(32) Molecule

Relationship Types

No relationships in database

Property Keys

Control Economic Environmental INCHI Legal

Added 32 labels, created 32 nodes, set 96 properties, completed after 2 ms.

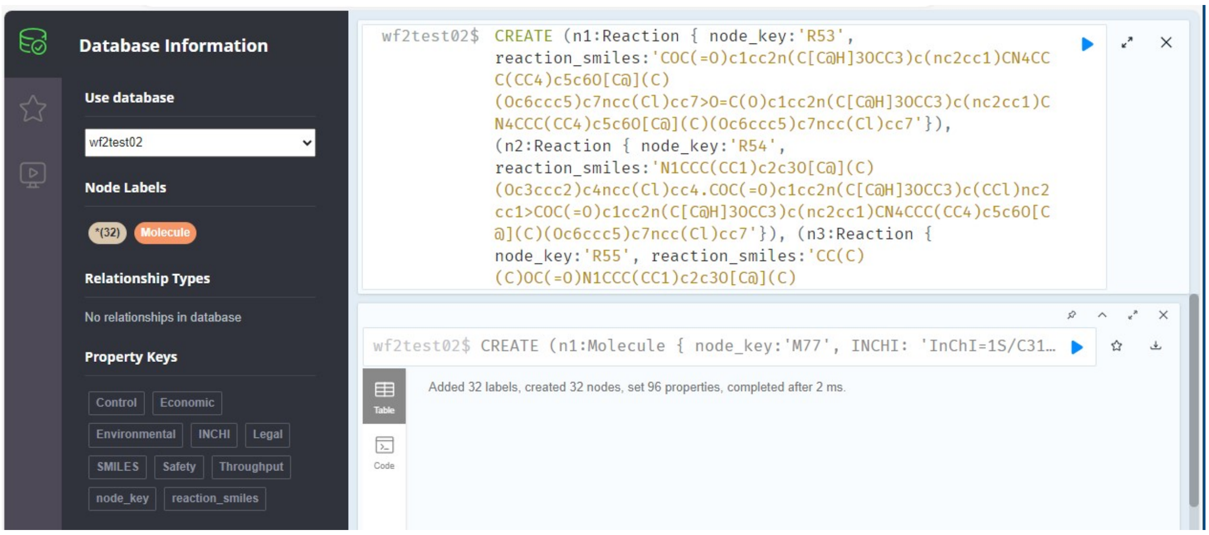
If at any point, it is required to start all over again, the following query can be used to delete all what has been saved:

```
>> MATCH (n) DETACH DELETE n
```

2. Query 2 to create all the Reaction nodes (implemented in the same way as query 1):

```
CREATE (n1:Reaction { node_key:'R53', reaction_smiles:'COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4
CCC(CC4)c5c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7>O=C(O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC
4)c5c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ',
Throughput:' '}),
(n2:Reaction { node_key:'R54', reaction_smiles:'N1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4.COC(
=O)c1cc2n(C[C@H]3OCC3)c(CCl)nc2cc1>COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC4)c5c
6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ',
Throughput:' '}),
(n3:Reaction { node_key:'R55', reaction_smiles:'CC(C)(C)OC(=O)N1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4
ncc(Cl)cc4>N1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '}),
(n4:Reaction { node_key:'R56', reaction_smiles:'CC(C)(C)OC(=O)N1CCC(=CC1)c2c3O[C@](C)(Oc3ccc2)
c4ncc(Cl)cc4>CC(C)(C)OC(=O)N1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4', Safety:' ',
Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n5:Reaction { node_key:'R57', reaction_smiles:'CC(C)(C)OC(=O)N1CCC(=CC1)B2OC(C)(C)C(C)(O2)C.C
lc1cnc(cc1)[C@]2(C)Oc3c(O2)cccc3Br>CC(C)(C)OC(=O)N1CCC(=CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(C
l)cc4', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n6:Reaction { node_key:'R58', reaction_smiles:'C#Cc1ncc(Cl)cc1.c1cc(Br)c(O)c(c1)O>Clc1cnc(cc1)[C@]2
(C)Oc3c(O2)cccc3Br', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n7:Reaction { node_key:'R59', reaction_smiles:'C1O[C@@H](C1)CNc2c(N)ccc(c2)C(=O)OC.COC(OC)(O
C)CCl>COC(=O)c1cc2n(C[C@H]3OCC3)c(CCl)nc2cc1', Safety:' ', Environmental:' ', Legal:' ', Economic:' ',
Control:' ', Throughput:' '}),
(n8:Reaction { node_key:'R60', reaction_smiles:'C1O[C@@H](C1)CNc2c(N(=O)=O)ccc(c2)C(=O)OC>C1O
[C@@H](C1)CNc2c(N)ccc(c2)C(=O)OC', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ',
Throughput:' '}), (n9:Reaction { node_key:'R61', reaction_smiles:'NC[C@H]1OCC1.[O-
][N+](=O)c1c(F)cc(cc1)C(=O)OC>C1O[C@@H](C1)CNc2c(N(=O)=O)ccc(c2)C(=O)OC', Safety:' ',
Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n10:Reaction { node_key:'R62', reaction_smiles:'CN(C)/C=C(Cl)/C=[N+](C)C.c1cc2O[C@](C)(C(=O)C)O
c2c(c1)Br>Clc1cnc(cc1)[C@]2(C)Oc3c(O2)cccc3Br', Safety:' ', Environmental:' ', Legal:' ', Economic:' ',
Control:' ', Throughput:' '}),
(n11:Reaction { node_key:'R63', reaction_smiles:'C1O[C@@H](C1)C(=O)Nc2c(N)ccc(c2)C(=O)OC>C1O[
C@@H](C1)CNc2c(N)ccc(c2)C(=O)OC', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ',
Throughput:' '}),
(n12:Reaction { node_key:'R64', reaction_smiles:'C1O[C@@H](C1)C(O)=O.Nc1c(N)cc(cc1)C(=O)OC>C1
O[C@@H](C1)C(=O)Nc2c(N)ccc(c2)C(=O)OC', Safety:' ', Environmental:' ', Legal:' ', Economic:' ',
Control:' ', Throughput:' '}),
(n13:Reaction { node_key:'R65', reaction_smiles:'COC(=O)c1cc2c(cc1)nc([nH]2)CCl.Cc1ccc(cc1)S(=O)(=O
)OC[C@H]2OCC2>COC(=O)c1cc2n(C[C@H]3OCC3)c(CCl)nc2cc1', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '}),
(n14:Reaction { node_key:'R66', reaction_smiles:'Nc1c(N)cc(cc1)C(=O)OC.COC(OC)(OC)CCl>COC(=O)c
1cc2c(cc1)nc([nH]2)CCl', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ', Throughput:' '}),
(n15:Reaction { node_key:'R67', reaction_smiles:'COC(=O)c1cc2n(C[C@H]3OCC3)c(C=O)nc2cc1.N1CCC(
CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4>COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC4)c5
c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ',
Throughput:' '}),
(n16:Reaction { node_key:'R68', reaction_smiles:'Fc1c(O)c(ccc1)C2CCN(CC2)Cc3n(C[C@H]4OCC4)c5c(n
3)ccc(c5)C(=O)OC.Cc1ccc(cc1)S(=O)(=O)O[C@@](C)(O)c2ncc(Cl)cc2>COC(=O)c1cc2n(C[C@H]3OCC3
)c(nc2cc1)CN4CCC(CC4)c5c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7', Safety:' ', Environmental:' ', Legal:' ',
Economic:' ', Control:' ', Throughput:' '}),
(n17:Reaction { node_key:'R69', reaction_smiles:'C1O[C@@H](C1)CNc2c(ccc(c2)C(OC)=O)NC(=O)CN3C
CC(CC3)c4c5O[C@](C)(Oc5ccc4)c6ncc(Cl)cc6>COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC
4)c5c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ',
Throughput:' '}),
(n18:Reaction { node_key:'R70', reaction_smiles:'NC[C@H]1OCC1.COC(=O)c1cc(F)c(cc1)NC(=O)CN2CC
C(CC2)c3c4O[C@](C)(Oc4ccc3)c5ncc(Cl)cc5>C1O[C@@H](C1)CNc2c(ccc(c2)C(OC)=O)NC(=O)CN3CC
C(CC3)c4c5O[C@](C)(Oc5ccc4)c6ncc(Cl)cc6', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:'
', Throughput:' '}),
(n19:Reaction { node_key:'R71', reaction_smiles:'O=C[C@H]1OCC1.COC(=O)c1cc(N)c(cc1)NC(=O)CN2C
CC(CC2)c3c4O[C@](C)(Oc4ccc3)c5ncc(Cl)cc5>C1O[C@@H](C1)CNc2c(ccc(c2)C(OC)=O)NC(=O)CN3C
CC(CC3)c4c5O[C@](C)(Oc5ccc4)c6ncc(Cl)cc6', Safety:' ', Environmental:' ', Legal:' ', Economic:' ',
Control:' ', Throughput:' '}),
(n20:Reaction { node_key:'R72', reaction_smiles:'C1O[C@@H](C1)CNc2c(N)ccc(c2)C(=O)OC.O=C(O)CN
1CCC(CC1)c2c3O[C@](C)(Oc3ccc2)c4ncc(Cl)cc4>C1O[C@@H](C1)CNc2c(ccc(c2)C(OC)=O)NC(=O)CN
3CCC(CC3)c4c5O[C@](C)(Oc5ccc4)c6ncc(Cl)cc6', Safety:' ', Environmental:' ', Legal:' ', Economic:' ',
Control:' ', Throughput:' '}),
(n21:Reaction { node_key:'R73', reaction_smiles:'COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CNCCC(CC=
O)c4c5O[C@](C)(Oc5ccc4)c6ncc(Cl)cc6>COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC4)c5c6
O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ',
Throughput:' '}),
(n22:Reaction { node_key:'R74', reaction_smiles:'COC(=O)c1cc2n(C[C@H]3OCC3)c(C=O)nc2cc1.NCCC(
CC=O)c1c2O[C@](C)(Oc2ccc1)c3ncc(Cl)cc3>COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CNCCC(CC=O
)c4c5O[C@](C)(Oc5ccc4)c6ncc(Cl)cc6', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:' ',
Throughput:' '})
```

Copying this query in neo4j browser will look like this:

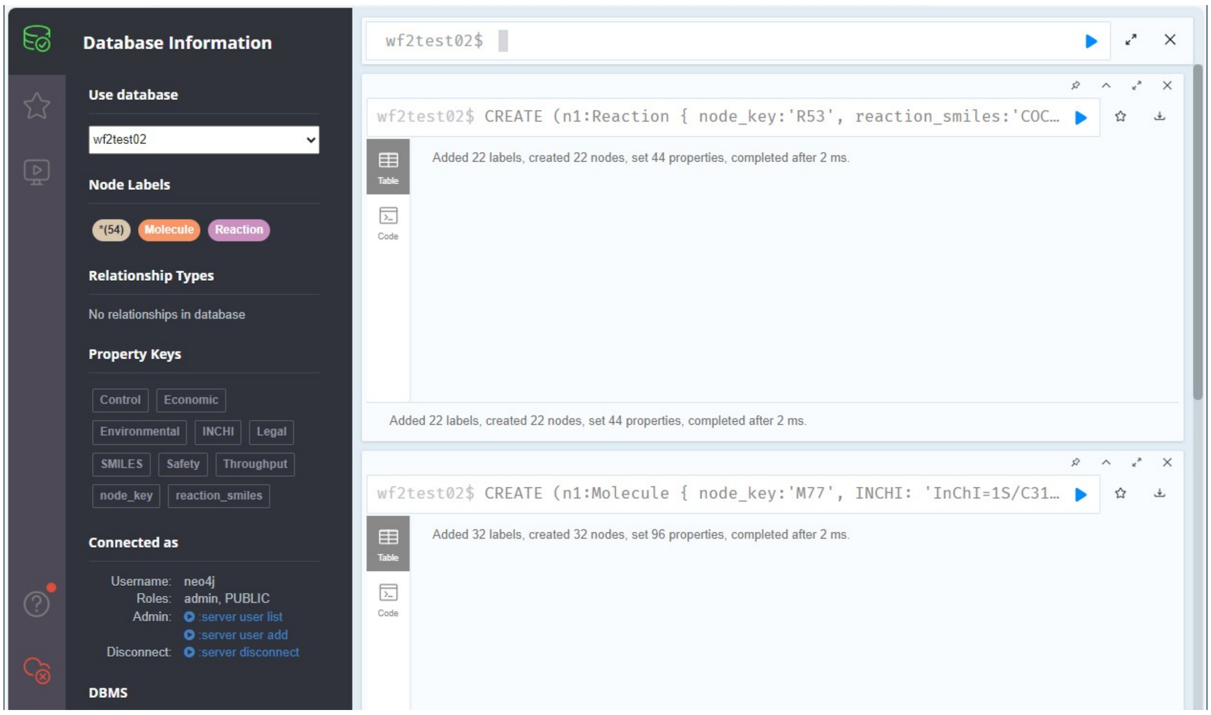


```
wf2test02$  CREATE (n1:Reaction { node_key:'R53',
            reaction_smiles:'COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CC
            C(CC4)c5c6O[C@](C)
            (Oc6ccc5)c7ncc(Cl)cc7>O=C(O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)C
            N4CCC(CC4)c5c6O[C@](C)(Oc6ccc5)c7ncc(Cl)cc7'}),
            (n2:Reaction { node_key:'R54',
            reaction_smiles:'N1CCC(CC1)c2c3O[C@](C)
            (Oc3ccc2)c4ncc(Cl)cc4.COC(=O)c1cc2n(C[C@H]3OCC3)c(CCl)nc2
            cc1>COC(=O)c1cc2n(C[C@H]3OCC3)c(nc2cc1)CN4CCC(CC4)c5c6O[C
            @](C)(Oc6ccc5)c7ncc(Cl)cc7'}), (n3:Reaction {
            node_key:'R55', reaction_smiles:'CC(C)
            (C)OC(=O)N1CCC(CC1)c2c3O[C@](C)
```

```
wf2test02$ CREATE (n1:Molecule { node_key:'M77', INCHI: 'InChI=1S/C31…
```

```
Added 32 labels, created 32 nodes, set 96 properties, completed after 2 ms.
```

Pressing the blue arrow at the top will run the query with the following answer:


Database Information

Use database

wf2test02

Node Labels

*(54) Molecule Reaction

Relationship Types

No relationships in database

Property Keys

Control Economic Environmental INCHI Legal SMILES Safety Throughput node_key reaction_smiles

Connected as

Username: neo4j
Roles: admin, PUBLIC
Admin: :server user list
:server user add
Disconnect: :server disconnect

DBMS

```
wf2test02$
```

```
wf2test02$ CREATE (n1:Reaction { node_key:'R53', reaction_smiles:'COC…
```

Added 22 labels, created 22 nodes, set 44 properties, completed after 2 ms.

Added 22 labels, created 22 nodes, set 44 properties, completed after 2 ms.

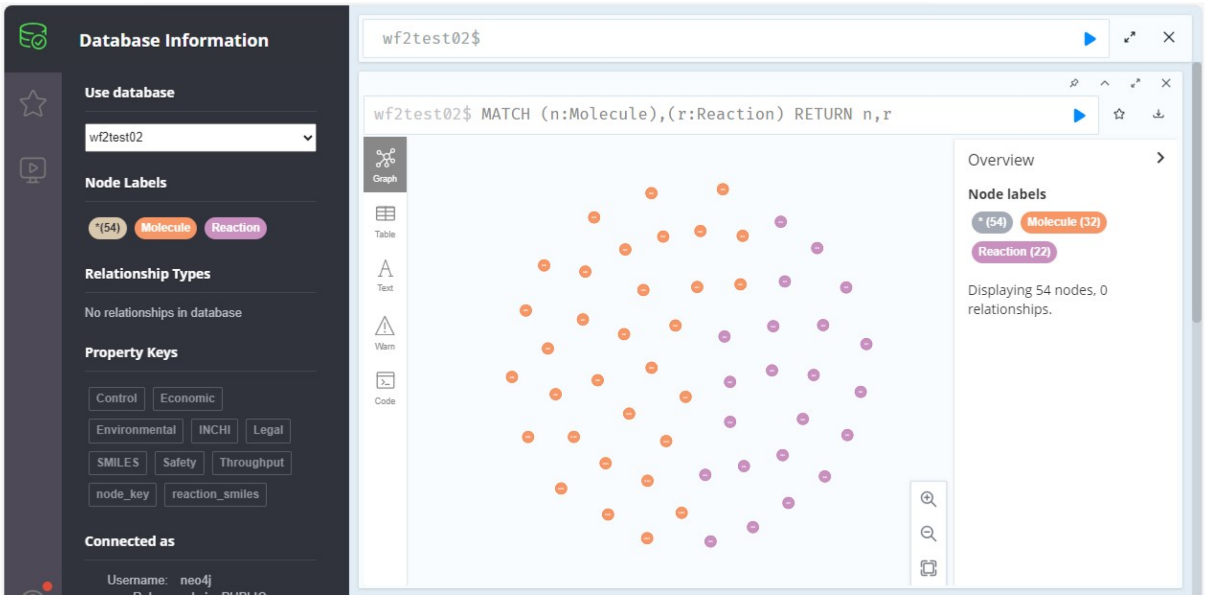
```
wf2test02$ CREATE (n1:Molecule { node_key:'M77', INCHI: 'InChI=1S/C31…
```

Added 32 labels, created 32 nodes, set 96 properties, completed after 2 ms.

A quick way to double check the creation of all the molecule and reaction nodes is running this short query:

```
MATCH (n:Molecule),(r:Reaction) RETURN n,r
```

Pressing the blue arrow at the top will run the query with the following answer:


Database Information

Use database

wf2test02

Node Labels

*(54) Molecule Reaction

Relationship Types

No relationships in database

Property Keys

Control Economic Environmental INCHI Legal SMILES Safety Throughput node_key reaction_smiles

Connected as

Username: neo4j

```
wf2test02$
```

```
wf2test02$ MATCH (n:Molecule),(r:Reaction) RETURN n,r
```

Overview

Node labels

* (54) Molecule (32) Reaction (22)

Displaying 54 nodes, 0 relationships.

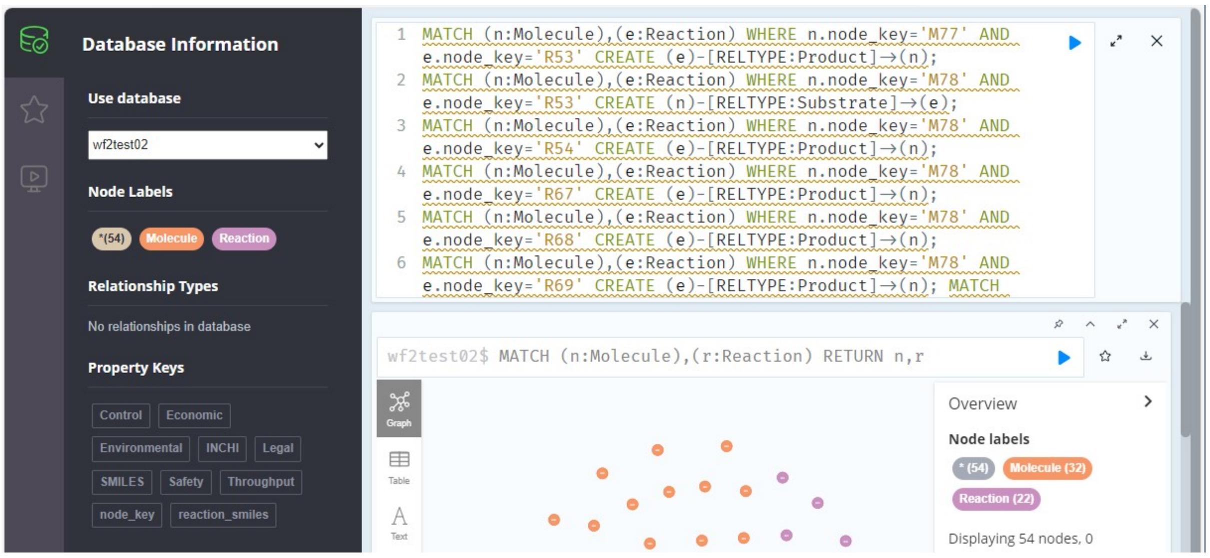
3. Query 3 to create all the relationships (substrate and product).

This query will execute one element at the time, wait 10s until all of them are completed.

```
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M77' AND e.node_key='R53' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND e.node_key='R53' CREATE (n)-
[RELTYPE:Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND e.node_key='R54' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND e.node_key='R67' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND e.node_key='R68' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND e.node_key='R69' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND e.node_key='R73' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M100' AND e.node_key='R68' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M101' AND e.node_key='R68' CREATE (n)-
[RELTYPE:Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M102' AND e.node_key='R69' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M79' AND e.node_key='R54' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M80' AND e.node_key='R54' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M79' AND e.node_key='R67' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M99' AND e.node_key='R67' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M107' AND e.node_key='R73' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M102' AND e.node_key='R70' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M102' AND e.node_key='R71' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M102' AND e.node_key='R72' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M80' AND e.node_key='R59' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M80' AND e.node_key='R65' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M79' AND e.node_key='R55' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M107' AND e.node_key='R74' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M104' AND e.node_key='R71' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M105' AND e.node_key='R71' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M103' AND e.node_key='R70' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M90' AND e.node_key='R70' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M106' AND e.node_key='R72' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M87' AND e.node_key='R72' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M88' AND e.node_key='R59' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M87' AND e.node_key='R59' CREATE (n)-
[RELTYPE: Substrate]->(e);
```

```
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M98' AND e.node_key='R65' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M97' AND e.node_key='R65' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M81' AND e.node_key='R55' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M99' AND e.node_key='R74' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M108' AND e.node_key='R74' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M90' AND e.node_key='R61' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M91' AND e.node_key='R61' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M89' AND e.node_key='R61' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M89' AND e.node_key='R60' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M87' AND e.node_key='R60' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M87' AND e.node_key='R63' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M94' AND e.node_key='R63' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M94' AND e.node_key='R64' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M95' AND e.node_key='R64' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M96' AND e.node_key='R64' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M96' AND e.node_key='R66' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M88' AND e.node_key='R66' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M97' AND e.node_key='R66' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M81' AND e.node_key='R56' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M82' AND e.node_key='R56' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M82' AND e.node_key='R57' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M84' AND e.node_key='R57' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M83' AND e.node_key='R57' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M83' AND e.node_key='R58' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M83' AND e.node_key='R62' CREATE (e)-
[RELTYPE:Product]->(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M93' AND e.node_key='R62' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M92' AND e.node_key='R62' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M85' AND e.node_key='R58' CREATE (n)-
[RELTYPE: Substrate]->(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M86' AND e.node_key='R58' CREATE (n)-
[RELTYPE: Substrate]->(e)
```

Copying this query in neo4j browser will look like this:

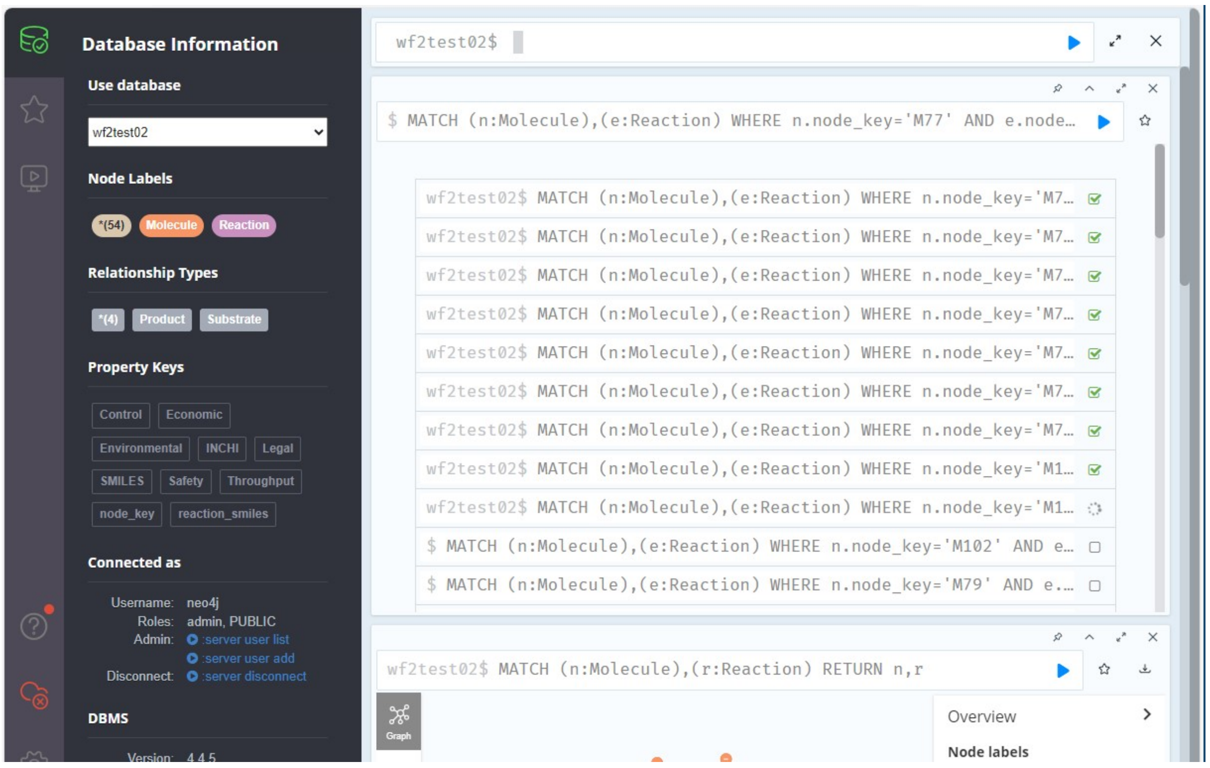


```
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M77' AND
e.node_key='R53' CREATE (e)-[RELTYPE:Product]→(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND
e.node_key='R53' CREATE (n)-[RELTYPE:Substrate]→(e);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND
e.node_key='R54' CREATE (e)-[RELTYPE:Product]→(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND
e.node_key='R67' CREATE (e)-[RELTYPE:Product]→(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND
e.node_key='R68' CREATE (e)-[RELTYPE:Product]→(n);
MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M78' AND
e.node_key='R69' CREATE (e)-[RELTYPE:Product]→(n); MATCH
```

```
wf2test02$ MATCH (n:Molecule),(r:Reaction) RETURN n,r
```

Displaying 54 nodes, 0

Pressing the blue arrow at the top will run the query with the following answer:



```
Username:  neo4j
Roles:  admin, PUBLIC
Admin:  :server user list
        :server user add
Disconnect:  :server disconnect
```

```
wf2test02$
```

```
$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M77' AND e.node…
```

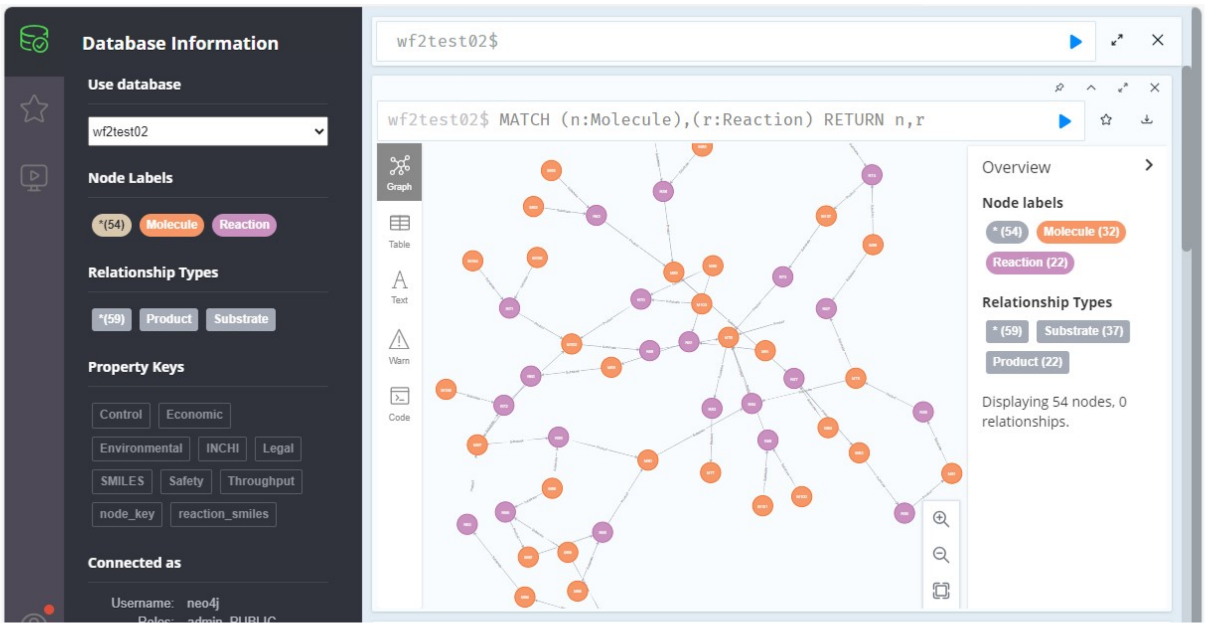
```
wf2test02$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M7…
wf2test02$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M7…
wf2test02$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M7…
wf2test02$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M7…
wf2test02$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M7…
wf2test02$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M7…
wf2test02$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M7…
wf2test02$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M1…
wf2test02$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M1…
$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M102' AND e…
$ MATCH (n:Molecule),(e:Reaction) WHERE n.node_key='M79' AND e.…
```

```
wf2test02$ MATCH (n:Molecule),(r:Reaction) RETURN n,r
```

Now we can see the whole network already created:

```
MATCH (n:Molecule),(r:Reaction) RETURN n,r
```

Pressing the blue arrow at the top will run the query with the following answer:



```
wf2test02$ MATCH (n:Molecule),(r:Reaction) RETURN n,r
```

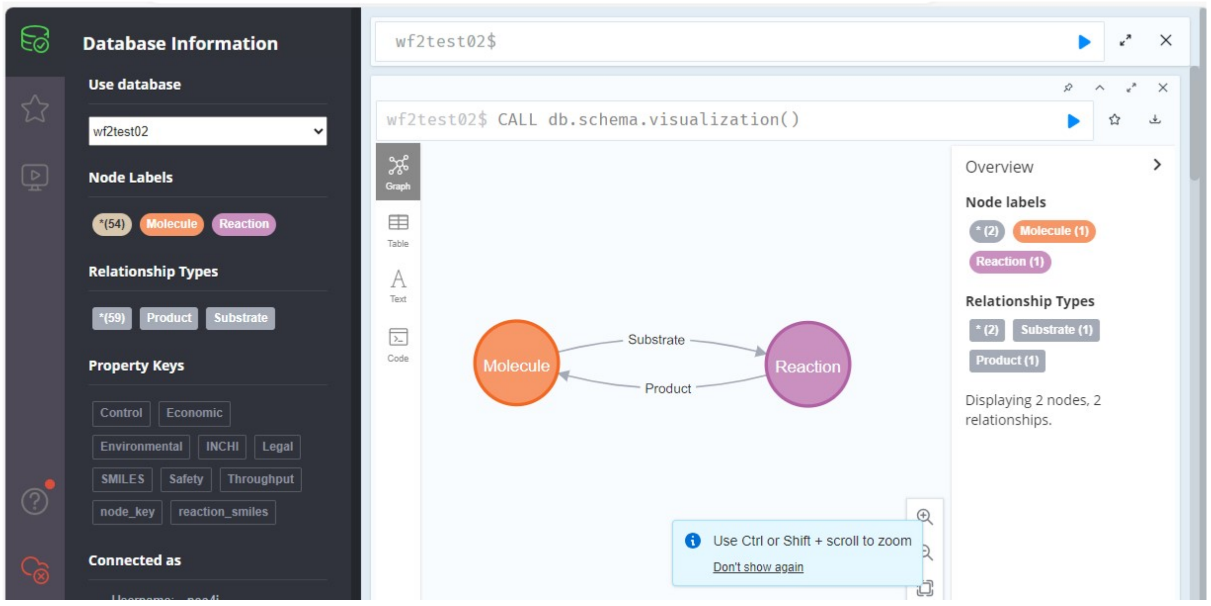
Displaying 54 nodes, 0 relationships.

In this window you can zoom and reorganise the nodes for a clearer visualisation. Nodes in Neo4j follow a gravity field, so if you pick a hub node and drag it around the canvas, the whole network will reorganise automatically.

Now the nodes contain all the relationships. To visualise the **data model** on the graph database (equivalent to a traditional Database Schema) run the following query:

```
CALL db.schema.visualization()
```

Pressing the blue arrow at the top will run the query with the following answer:



```
wf2test02$ CALL db.schema.visualization()
```

Displaying 2 nodes, 2 relationships.

Use Ctrl or Shift + scroll to zoom

Route nodes

The route nodes illustrated on the paper are created automatically using an algorithm that identify all the routes from the entire network. In this case, as we are not providing access to this specific algorithm, the number of routes nodes already identified (12) will be used to create them on the network. To achieve this, two queries are needed:

To create the 12 Route nodes run this query:

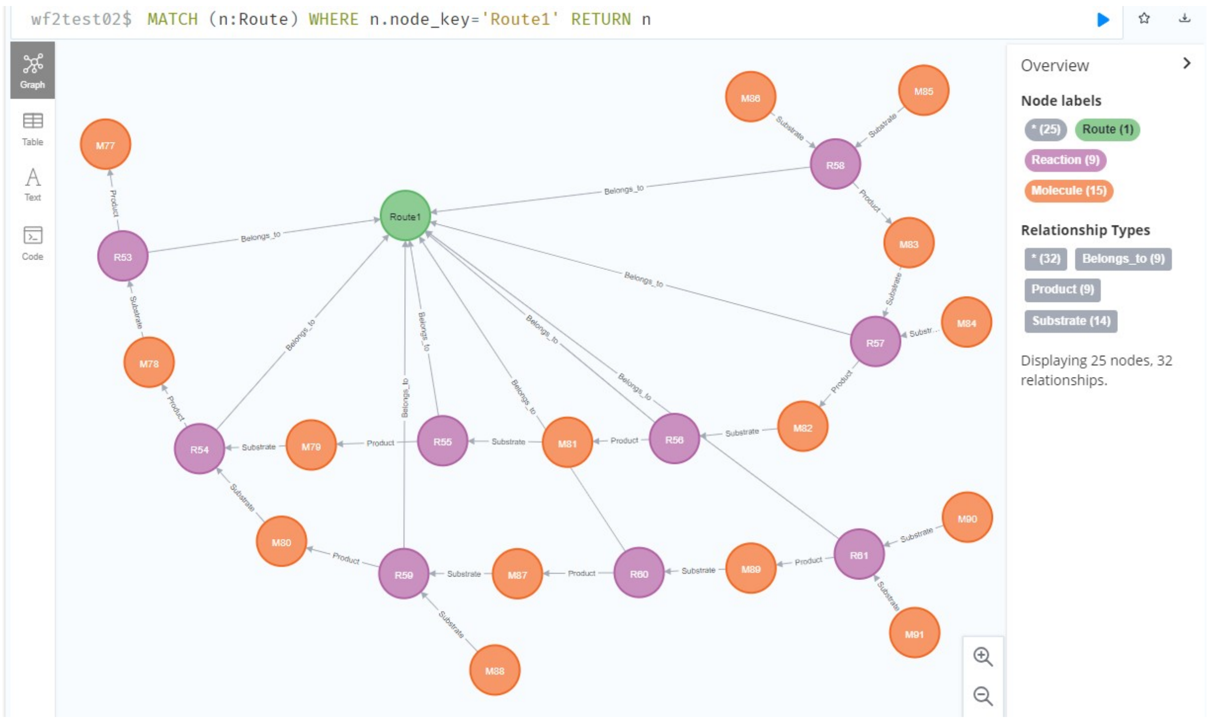
```
CREATE (n1:Route { node_key:'Route1', Safety:'   ', Environmental:'   ', Legal:'   ', Economic:'   ', Control:'   ',
Throughput:' '}), (n2:Route { node_key:'Route2', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:'
', Throughput:' '}), (n3:Route { node_key:'Route3', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:'
', Throughput:' '}), (n4:Route { node_key:'Route4', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:'
', Throughput:' '}), (n5:Route { node_key:'Route5', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:'
', Throughput:' '}), (n6:Route { node_key:'Route6', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:'
', Throughput:' '}), (n7:Route { node_key:'Route7', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:'
', Throughput:' '}), (n8:Route { node_key:'Route8', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:'
', Throughput:' '}), (n9:Route { node_key:'Route9', Safety:' ', Environmental:' ', Legal:' ', Economic:' ', Control:'
',   Throughput:'   '}),   (n10:Route { node_key:'Route10', Safety:'   ', Environmental:'   ', Legal:'   ', Economic:'
', Control:'   ',   Throughput:'   '}),   (n11:Route { node_key:'Route11', Safety:'   ', Environmental:'   ', Legal:'
', Economic:'   ', Control:'   ',   Throughput:'   '}),   (n12:Route { node_key:'Route12', Safety:'   ', Environmental:'
', Legal:' ', Economic:' ', Control:' ', Throughput:' '})
```

To create the relationships run these queries:

Route 1

```
MATCH (n:Route),(e1:Reaction),(e2:Reaction),(e3:Reaction),(e4:Reaction),(e5:Reaction),(e6:Reaction),(e7:
Reaction),(e8:Reaction),(e9:Reaction) WHERE n.node_key='Route1' AND e1.node_key='R53' AND e2.node
_key='R54' AND e3.node_key='R59' AND e4.node_key='R55'
AND e5.node_key='R60' AND e6.node_key='R56' AND e7.node_key='R61' AND e8.node_key='R57'
AND e9.node_key='R58' CREATE (e1)-[REL1:Belongs_to]->(n), (e2)-[REL2:Belongs_to]->(n),  (e3)-
[REL3:Belongs_to]->(n), (e4)-[REL4:Belongs_to]->(n),  (e5)-[REL5:Belongs_to]->(n), (e6)-
[REL6:Belongs_to]->(n), (e7)-[REL7:Belongs_to]->(n), (e8)-[REL8:Belongs_to]->(n), (e9)-
[REL9:Belongs_to]->(n)
```

The resulting route can be visualised by unhiding all the adjacent nodes resulting on this:



```
wf2test02$ MATCH (n:Route) WHERE n.node_key='Route1' RETURN n
```

Overview

Node labels

* (25) Route (1) Reaction (9) Molecule (15)

Relationship Types

* (32) Belongs_to (9) Product (9) Substrate (14)

Displaying 25 nodes, 32 relationships.

In the same way, to get all the other routes, run the query below.

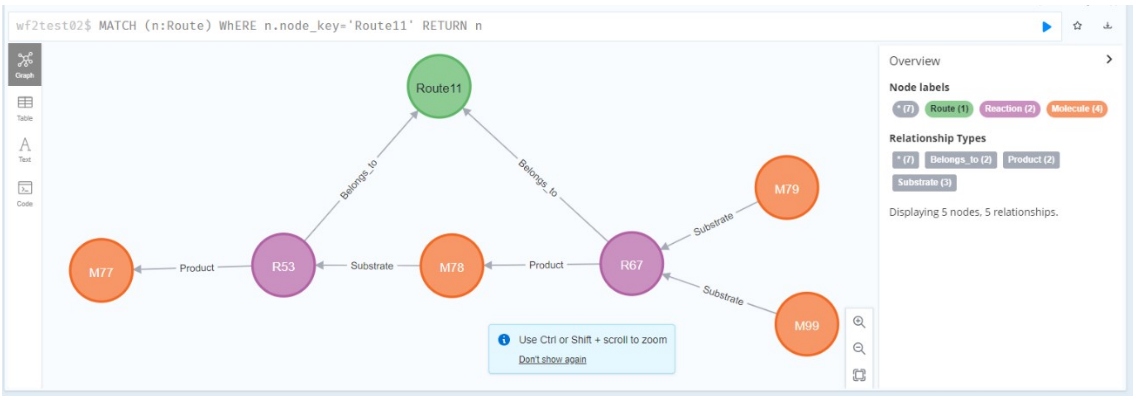
Route 2 to 10 – copy and paste whole block

```
MATCH (n:Route),(e1:Reaction),(e2:Reaction),(e3:Reaction),(e4:Reaction),(e5:Reaction),(e6:Reaction),(e7:
Reaction),(e8:Reaction),(e9:Reaction)
WHERE n.node_key='Route2' AND e1.node_key='R53' AND e2.node_key='R54'
AND e3.node_key='R59' AND e4.node_key='R55' AND e5.node_key='R63' AND e6.node_key='R56'
AND e7.node_key='R64' AND e8.node_key='R57' AND e9.node_key='R58'
CREATE (e1)-[REL1:Belongs_to]->(n), (e2)-[REL2:Belongs_to]->(n), (e3)-[REL3:Belongs_to]->(n), (e4)-
[REL4:Belongs_to]->(n), (e5)-[REL5:Belongs_to]->(n), (e6)-[REL6:Belongs_to]->(n), (e7)-
[REL7:Belongs_to]->(n), (e8)-[REL8:Belongs_to]->(n), (e9)-[REL9:Belongs_to]->(n);
MATCH (n:Route),(e1:Reaction),(e2:Reaction),(e3:Reaction),(e4:Reaction),(e5:Reaction),(e6:Reaction),(e7:
Reaction),(e8:Reaction),(e9:Reaction)
WHERE n.node_key='Route3' AND e1.node_key='R53' AND e2.node_key='R54'
AND e3.node_key='R59' AND e4.node_key='R55' AND e5.node_key='R60' AND e6.node_key='R56'
AND e7.node_key='R61' AND e8.node_key='R57' AND e9.node_key='R62'
CREATE (e1)-[REL1:Belongs_to]->(n), (e2)-[REL2:Belongs_to]->(n), (e3)-[REL3:Belongs_to]->(n), (e4)-
[REL4:Belongs_to]->(n), (e5)-[REL5:Belongs_to]->(n), (e6)-[REL6:Belongs_to]->(n), (e7)-
[REL7:Belongs_to]->(n), (e8)-[REL8:Belongs_to]->(n), (e9)-[REL9:Belongs_to]->(n);
MATCH (n:Route),(e1:Reaction),(e2:Reaction),(e3:Reaction),(e4:Reaction),(e5:Reaction),(e6:Reaction),(e7:
Reaction),(e8:Reaction),(e9:Reaction)
WHERE n.node_key='Route4' AND e1.node_key='R53' AND e2.node_key='R54'
AND e3.node_key='R59' AND e4.node_key='R55' AND e5.node_key='R63' AND e6.node_key='R56'
AND e7.node_key='R64' AND e8.node_key='R57' AND e9.node_key='R62'
CREATE (e1)-[REL1:Belongs_to]->(n), (e2)-[REL2:Belongs_to]->(n), (e3)-[REL3:Belongs_to]->(n), (e4)-
[REL4:Belongs_to]->(n), (e5)-[REL5:Belongs_to]->(n), (e6)-[REL6:Belongs_to]->(n), (e7)-
[REL7:Belongs_to]->(n), (e8)-[REL8:Belongs_to]->(n), (e9)-[REL9:Belongs_to]->(n);
MATCH (n:Route),(e1:Reaction),(e2:Reaction),(e3:Reaction),(e4:Reaction),(e5:Reaction),(e6:Reaction),(e7:
Reaction),(e8:Reaction)      WHERE n.node_key='Route5' AND e1.node_key='R53' AND e2.node_key='R54'
AND e3.node_key='R65' AND e4.node_key='R55'       AND e5.node_key='R66' AND e6.node_key='R56'
AND e7.node_key='R57' AND e8.node_key='R58'       CREATE (e1)-[REL1:Belongs_to]->(n),        (e2)-
[REL2:Belongs_to]->(n), (e3)-[REL3:Belongs_to]->(n),        (e4)-[REL4:Belongs_to]->(n),        (e5)-
[REL5:Belongs_to]->(n),        (e6)-[REL6:Belongs_to]->(n),        (e7)-[REL7:Belongs_to]->(n),        (e8)-
[REL8:Belongs_to]->(n);
MATCH (n:Route),(e1:Reaction),(e2:Reaction),(e3:Reaction),(e4:Reaction),(e5:Reaction),(e6:Reaction),(e7:
```

```
Reaction),(e8:Reaction)     WHERE n.node_key='Route6' AND e1.node_key='R53' AND e2.node_key='R54'
AND e3.node_key='R65' AND e4.node_key='R55'          AND e5.node_key='R66' AND e6.node_key='R56'
AND e7.node_key='R57' AND e8.node_key='R62'        CREATE (e1)-[REL1:Belongs_to]->(n),        (e2)-
[REL2:Belongs_to]->(n),        (e3)-[REL3:Belongs_to]->(n),        (e4)-[REL4:Belongs_to]->(n),        (e5)-
[REL5:Belongs_to]->(n),        (e6)-[REL6:Belongs_to]->(n),        (e7)-[REL7:Belongs_to]->(n),        (e8)-
[REL8:Belongs_to]->(n);
MATCH (n:Route),(e1:Reaction),(e2:Reaction),(e3:Reaction) WHERE n.node_key='Route7' AND e1.node_
key='R53' AND e2.node_key='R69'  AND e3.node_key='R71'  CREATE (e1)-[REL1:Belongs_to]->(n),  (e2)-
[REL2:Belongs_to]->(n) , (e3)-[REL3:Belongs_to]->(n);
MATCH (n:Route),(e1:Reaction),(e2:Reaction),(e3:Reaction) WHERE n.node_key='Route8' AND e1.node_
key='R53' AND e2.node_key='R69'  AND e3.node_key='R70'  CREATE (e1)-[REL1:Belongs_to]->(n),  (e2)-
[REL2:Belongs_to]->(n) , (e3)-[REL3:Belongs_to]->(n);
MATCH (n:Route),(e1:Reaction),(e2:Reaction),(e3:Reaction) WHERE n.node_key='Route9' AND e1.node_
key='R53' AND e2.node_key='R69'  AND e3.node_key='R72'  CREATE (e1)-[REL1:Belongs_to]->(n),  (e2)-
[REL2:Belongs_to]->(n) , (e3)-[REL3:Belongs_to]->(n);
MATCH (n:Route),(e1:Reaction),(e2:Reaction),(e3:Reaction) WHERE n.node_key='Route10' AND e1.node
_key='R53' AND e2.node_key='R73' AND e3.node_key='R74' CREATE (e1)-[REL1:Belongs_to]->(n), (e2)-
[REL2:Belongs_to]->(n) , (e3)-[REL3:Belongs_to]->(n);
MATCH (n:Route),(e1:Reaction),(e2:Reaction)  WHERE n.node_key='Route11' AND e1.node_key='R53' A
ND e2.node_key='R67' CREATE (e1)-[REL1:Belongs_to]->(n), (e2)-[REL2:Belongs_to]->(n);
MATCH (n:Route),(e1:Reaction),(e2:Reaction)  WHERE n.node_key='Route12' AND e1.node_key='R53' A
ND e2.node_key='R68' CREATE (e1)-[REL1:Belongs_to]->(n), (e2)-[REL2:Belongs_to]->(n)
```

And two extra examples of the resulting routes:

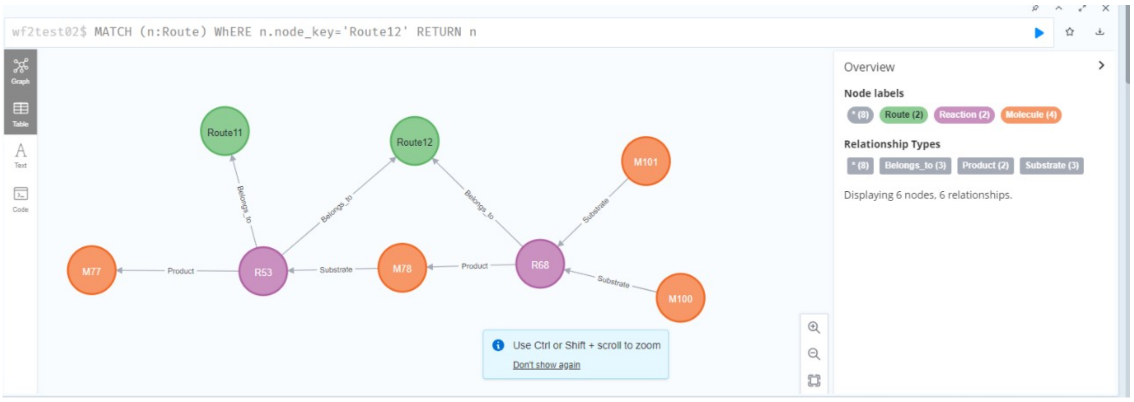
Route 11



```
wf2test02$ MATCH (n:Route) WHERE n.node_key='Route11' RETURN n
```

Displaying 5 nodes, 5 relationships.

Route 12



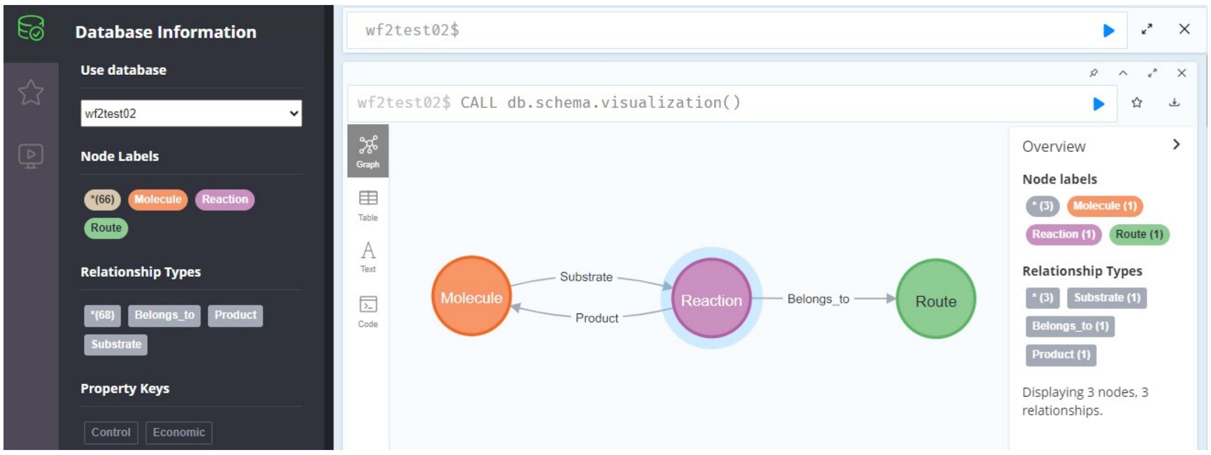
```
wf2test02$ MATCH (n:Route) WHERE n.node_key='Route12' RETURN n
```

Displaying 6 nodes, 6 relationships.

Finally, to visualise the **data model** again on the graph database (after introducing the route node) run the following query:

```
CALL db.schema.visualization()
```

Pressing the blue arrow at the top will run the query with the following answer:



Database Information

Use database: wf2test02

Node Labels: *(66) Molecule Reaction Route

Relationship Types: *(68) Belongs_to Product Substrate

Property Keys: Control Economic

```
wf2test02$ CALL db.schema.visualization()
```

Overview

Node labels: * (3) Molecule (1) Reaction (1) Route (1)

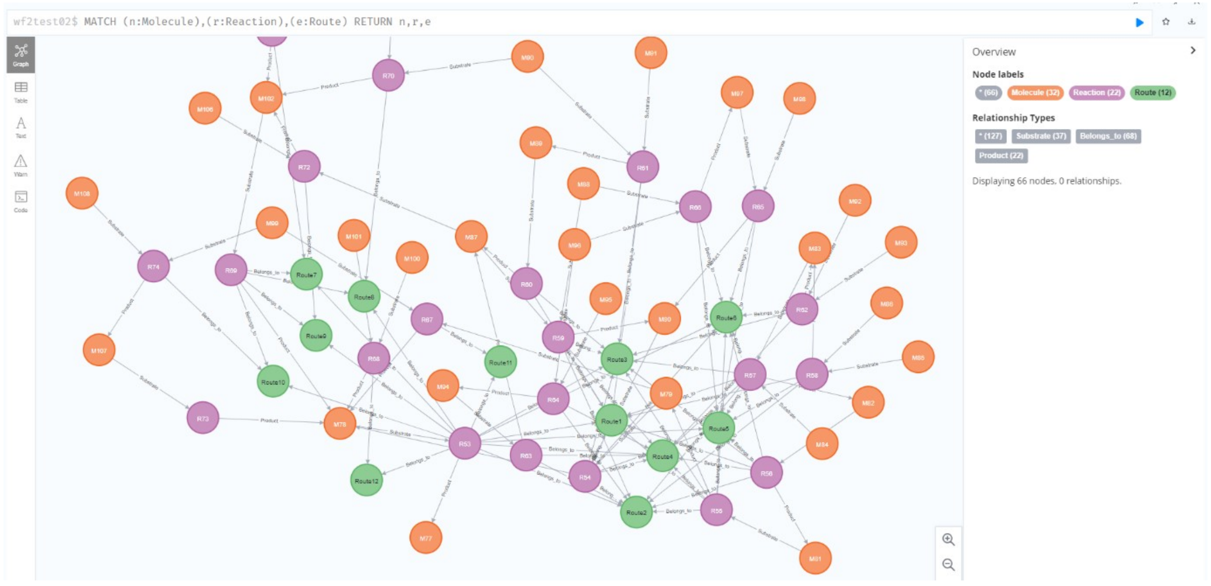
Relationship Types: * (3) Substrate (1) Belongs_to (1) Product (1)

Displaying 3 nodes, 3 relationships.

Now the database contains an intricated array of connections and browsing manually become difficult. For this reason, queries to uncover specific parts of the graph are needed.

For instance, this query will reveal the whole graph (this might take time to be resolved):

```
MATCH (n:Molecule),(r:Reaction),(e:Route) RETURN n,r,e
```



```
wf2test02$ MATCH (n:Molecule),(r:Reaction),(e:Route) RETURN n,r,e
```

Overview

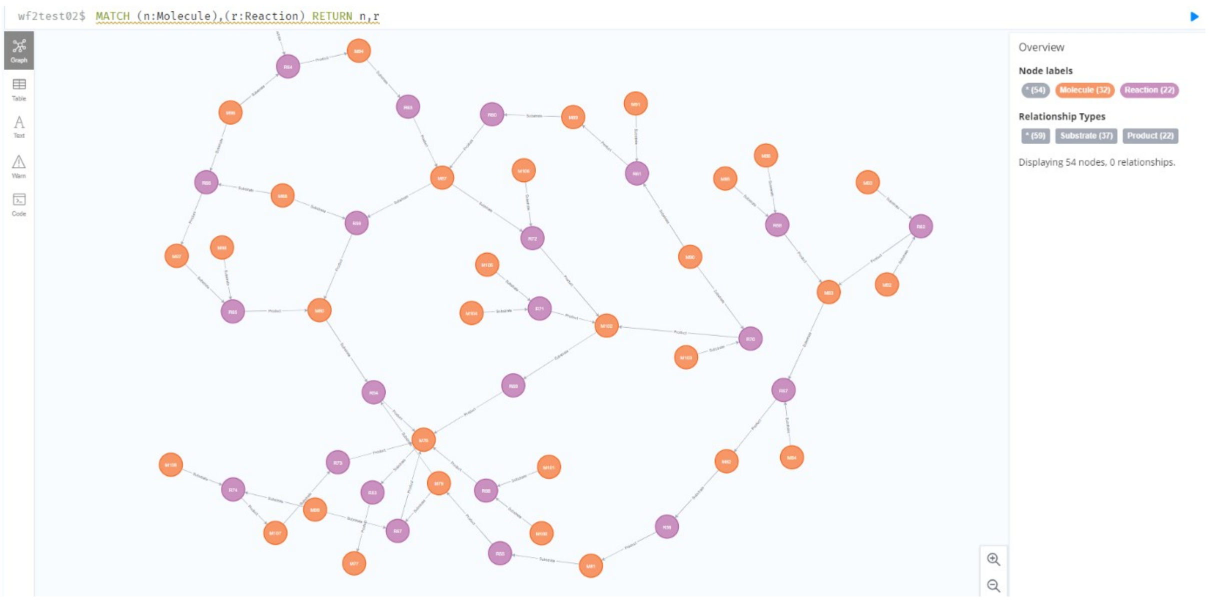
Node labels: * (66) Molecule (32) Reaction (22) Route (12)

Relationship Types: * (127) Substrate (37) Belongs_to (68) Product (22)

Displaying 66 nodes, 0 relationships.

For instance, reducing the query will return the reactions and molecule nodes only:

```
MATCH (n:Molecule),(r:Reaction) RETURN n,r
```



```
wf2test02$ MATCH (n:Molecule),(r:Reaction) RETURN n,r
```

Overview

Node labels

* (54) Molecule (32) Reaction (22)

Relationship Types

* (59) Substrate (37) Product (22)

Displaying 54 nodes, 0 relationships.