# A Software Package for Chemically Inspired Graph Transformation

DM840 - Week 39

Molecules as Labelled Graphs

- ▶ **An old idea**: [J. J. Sylvester, *Chemistry and Algebra*, Nature 1878]
- ▶ **Molecule**: simple, connected, labelled graph.
- ▶ **Vertex labels**: atom type, charge.
- ▶ **Edge labels**: bond type.
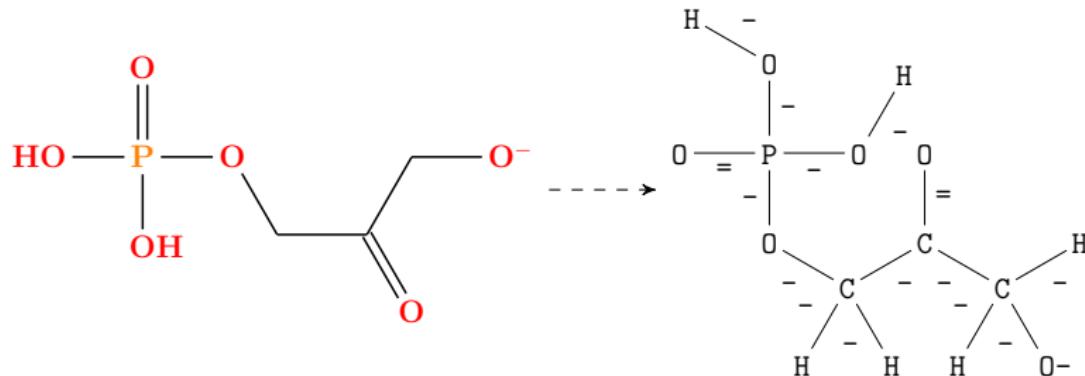
Simple: 2 single bonds is not the same as 1 double bond.
Connected: all atom interactions must be encoded as edges.

Using MØD

- Python script: "test.py":

```
myGraph = graphGMLString("""
  graph [
    node [ id 0 label "u" ]
    node [ id 1 label "v" ]
    edge [ source 0 target 1 label "e" ]
  ]""")
myGraph.print()
```

- Run the wrapper script:

```
mod -f test.py
```

enhanced version of

```
python3 test.py
mod_post
```

- Look at the generated "summary/summary.pdf".

External Graph Representaion

GML: list of vertices and edges.

SMILES: Simplified Molecular-Input Line-entry System

- ▶ Recording of a depth-first traversal.
- ▶ Only for molecules.
- ▶ Most hydrogen atoms are implicit.

GraphDFS: similar to SMILES but for general graphs.

```
ethanol1 = smiles("CCO", name="Ethanol1")
ethanol2 = graphDFS("[C]([H])([H])([H])[C]([H])([H])[O][H]", name="Ethanol2")
ethanol3 = graphDFS("CCO", name="Ethanol3")
ethanol4 = graphGMLString("""graph [
    node [ id 0 label "C" ]    node [ id 1 label "C" ]    node [ id 2 label "O" ]
    node [ id 3 label "H" ]    node [ id 4 label "H" ]    node [ id 5 label "H" ]
    node [ id 6 label "H" ]    node [ id 7 label "H" ]    node [ id 8 label "H" ]
    edge [ source 1 target 0 label "-" ]    edge [ source 2 target 1 label "-" ]
    edge [ source 3 target 0 label "-" ]    edge [ source 4 target 0 label "-" ]
    edge [ source 5 target 0 label "-" ]    edge [ source 6 target 1 label "-" ]
    edge [ source 7 target 1 label "-" ]    edge [ source 8 target 2 label "-" ]
]""", name="Ethanol4")
```

Reactions as Graph Transformations
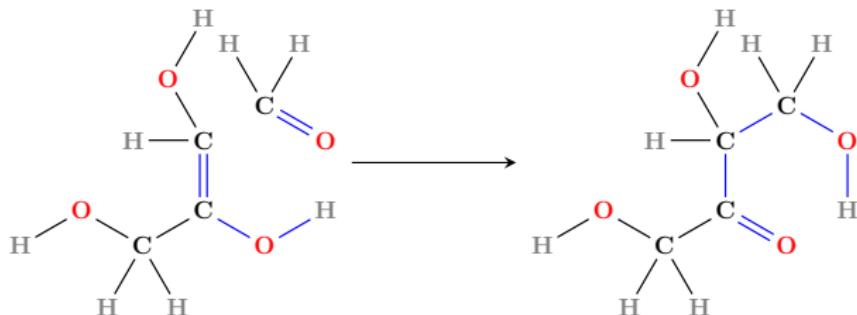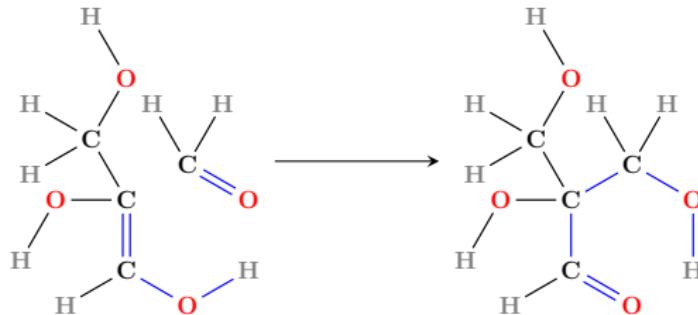
(Chemical constraint: only bonds and charges may change.)

Reactions as Double Pushout Transformations

$$p = (L \xleftarrow{l} K \xrightarrow{r} R) \qquad G \xRightarrow{p,m} H$$
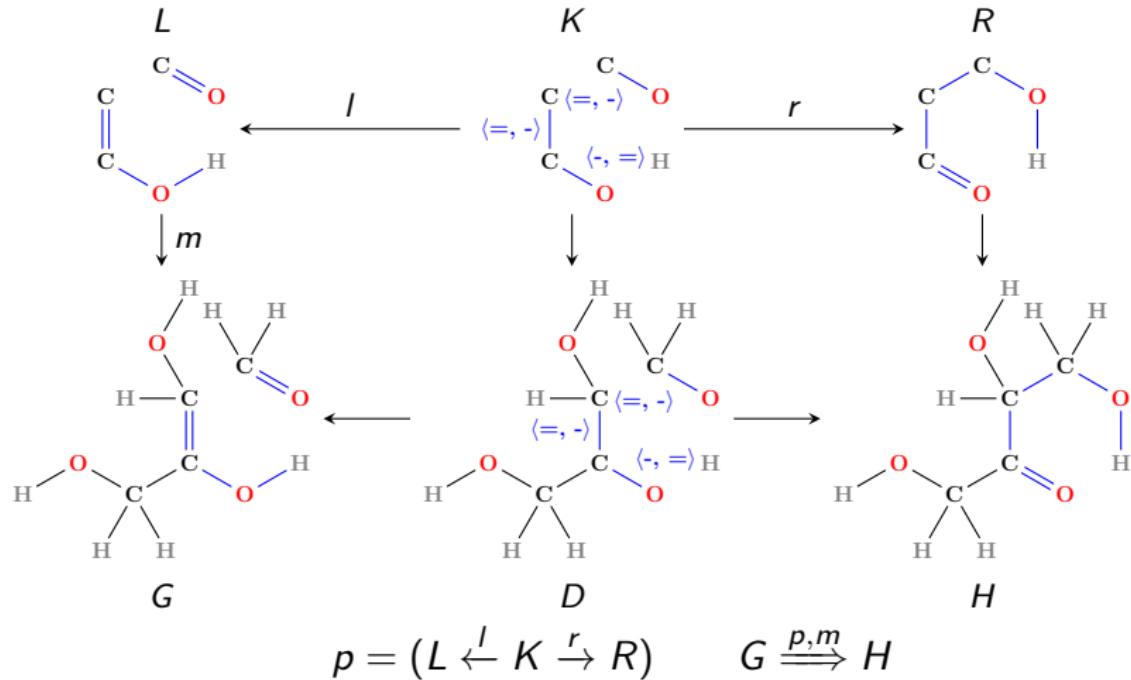
Restriction: only injective morphisms.

Pushouts and Simple Graphs
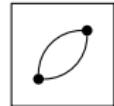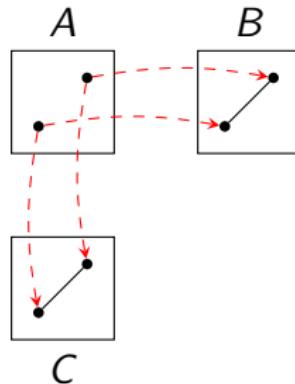
What is the pushout?

 ? No, parallel edges not allowed.

# Pushouts and Simple Graphs
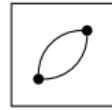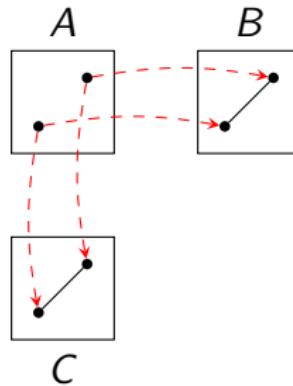
What is the pushout?

? No, parallel edges not allowed.

? No, merging of two chemical bonds not meaningful.

Pushouts and Simple Graphs

What is the pushout?
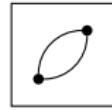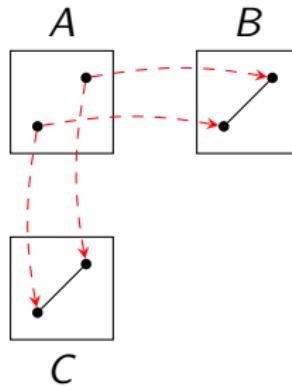
? No, parallel edges not allowed.

? No, merging of two chemical bonds not meaningful.

The pushout does not exist.

External Rule Representation

```
destroyVertex = ruleGMLString("""rule [
    left [
        node [ id 1 label "A" ]
    ]
]""")
createVertex = ruleGMLString("""rule [
    right [
        node [ id 1 label "A" ]
    ]
]""")
identity = ruleGMLString("""rule [
    context [
        node [ id 1 label "A" ]
    ]
]""")
labelChange = ruleGMLString("""rule [
    left [
        node [ id 1 label "A" ]
        edge [ source 1 target 2 label "A" ]
    ]
    context [
        node [ id 2 label "Q" ]
    ]
    right [
        node [ id 1 label "B" ]
        edge [ source 1 target 2 label "B" ]
    ]
]""")
```

$L =$ 'left' $\cup$ 'context' $R =$ 'right' $\cup$ 'context'

$K =$ 'context' $\cup$ ('left' $\cap$ 'right')

Transformation of Multisets of Connected Graphs

$$\{\!\{g_0, g_1\}\!\} \xRightarrow{p,m} \{\!\{h_0\}\!\}$$
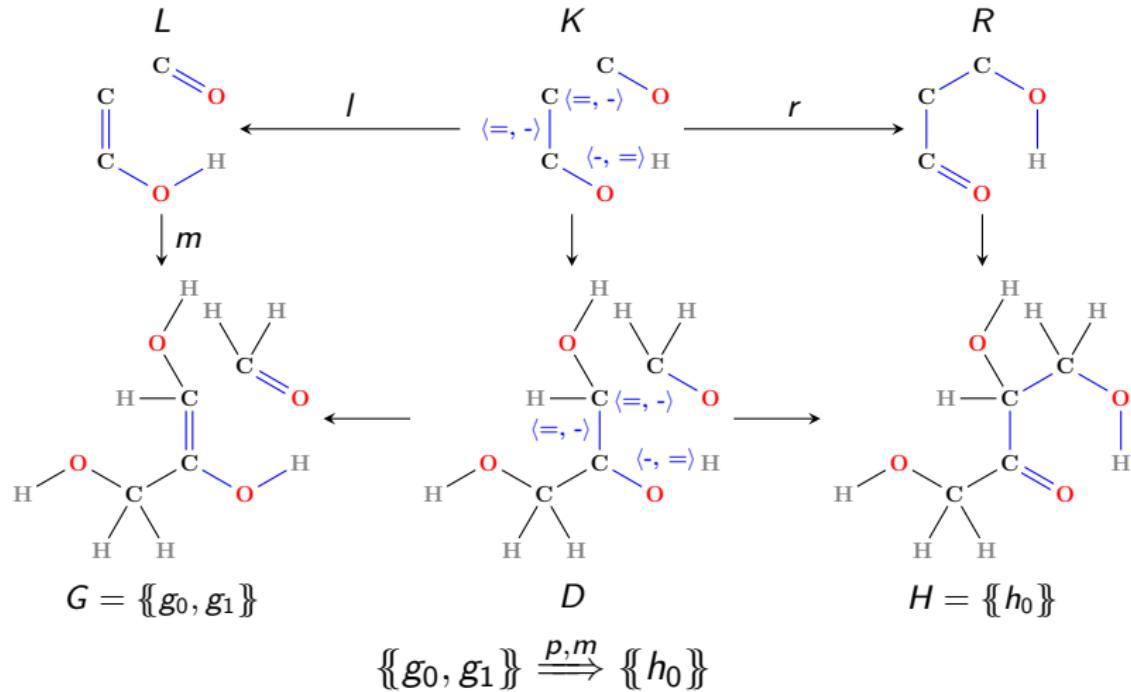
In general reactions transform multisets of molecules.

Chemistries as Graph Grammars

A chemistry:

- A set of initial molecules (graphs) $\mathcal{G}$.
- A set of reaction rules (DPO rules) $\mathcal{P}$.

Which molecules (graphs) can be reached?

Which reactions (direct derivations) can happen?

Chemistries as Graph Grammars

A chemistry:

- ▶ A set of initial molecules (graphs) $\mathcal{G}$.
- ▶ A set of reaction rules (DPO rules) $\mathcal{P}$.

Which molecules (graphs) can be reached?
Which reactions (direct derivations) can happen?

Derivation Graph:

- ▶ A directed multi-hypergraph.
- ▶ Vertices are connected graphs.
- ▶ Hyperedges are direct derivations.
- ▶ Can model a chemical reaction network.
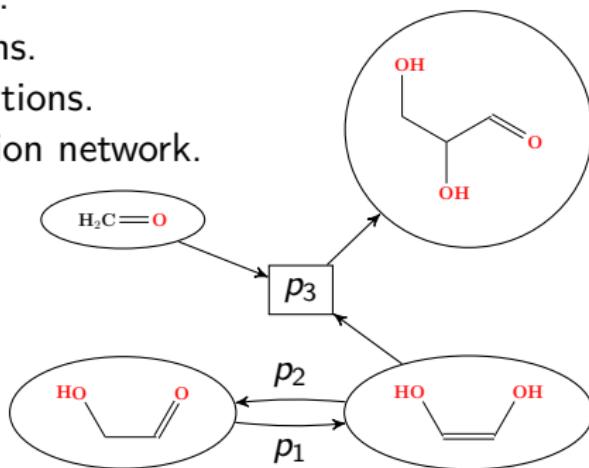
Rule Application

Given: a set of graphs $\mathcal{U}$, and a rule $p = (L \leftarrow K \rightarrow R)$.

Brute-force:

1. Enumerate all multisets based on $\mathcal{U}$.
2. Try applying p to each of them.
3. For each successful direct derivation $G \stackrel{p}{\Rightarrow} H$, split H into connected components.
4. Let $\mathcal{U}' = \mathcal{U} \cup \langle \text{the new graphs} \rangle$.

Strategic Exploration of Graph Languages

An embedded domain-specific programming language.

State: a set of graphs, with a distinguished subset.

Instructions:

- 'addSubset(g)': add 'g' to the state.
- 'filterSubset(predicate)': remove graphs from the state.
- 'p': apply the rule 'p', and add the new graphs.
- 'q0 >> q1': sequence, first do 'q0' then 'q1'.
- '[q0, q1]': parallel, do 'q0' and 'q1' independently.
- 'repeat[k](q)': sequence 'q' at most 'k' times.
- 'repeat(q)': stop when no new graphs can be found.
- ...

Example:

```
strat = addSubset(inputGraphs) >> repeat(inputRules)
```

Composition of Transformation Rules

Given: rules $p_1 = (L_1 \leftarrow K_1 \rightarrow R_1)$ and $p_2 = (L_2 \leftarrow K_2 \rightarrow R_2)$.

What if R_1 is a supergraph of L_2?

Composition of Transformation Rules

Given: rules $p_1 = (L_1 \leftarrow K_1 \rightarrow R_1)$ and $p_2 = (L_2 \leftarrow K_2 \rightarrow R_2)$.

What if R_1 is a supergraph of L_2?
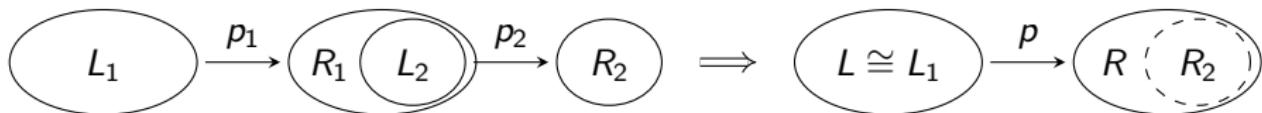
Composition of Transformation Rules

Given: rules $p_1 = (L_1 \leftarrow K_1 \rightarrow R_1)$ and $p_2 = (L_2 \leftarrow K_2 \rightarrow R_2)$.

What if R_1 is a supergraph of L_2?
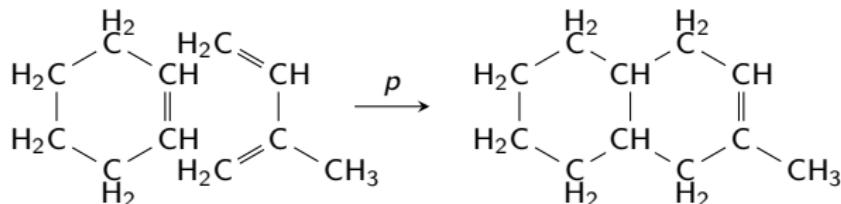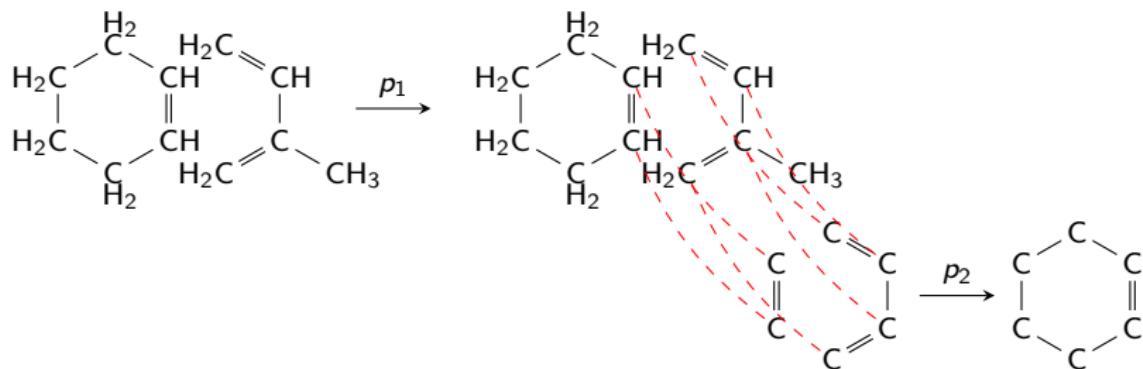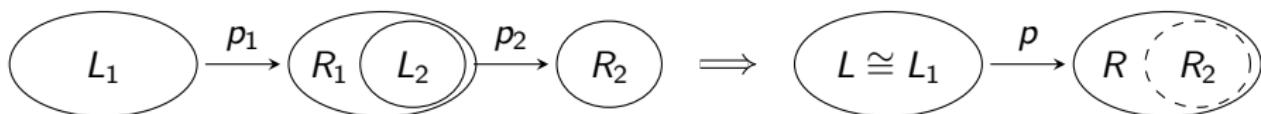
Partial Rule Composition
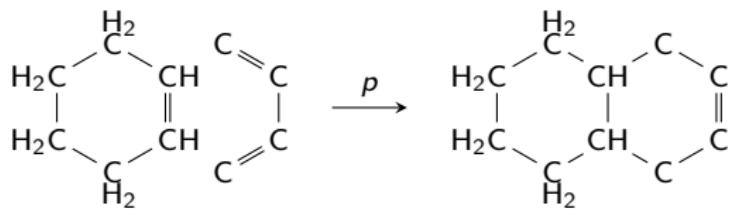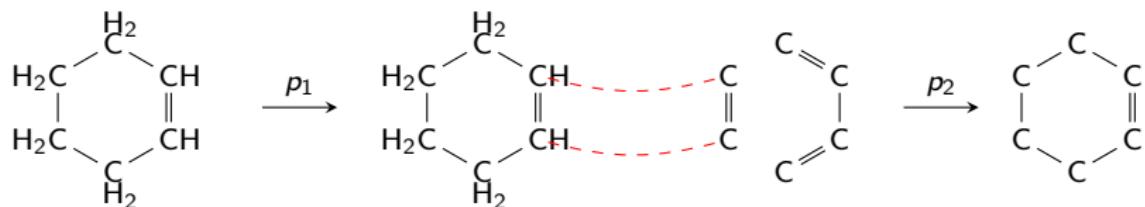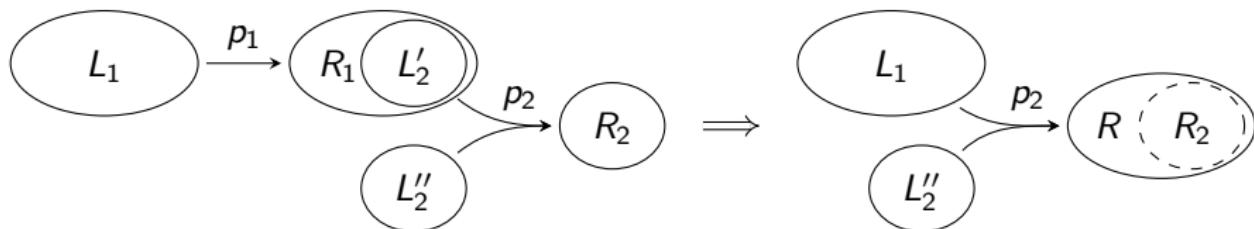
Graphs as Rules

Identity Rule: $(G \leftarrow G \rightarrow G)$
"match G and do nothing"

Bind Rule: $(\emptyset \leftarrow \emptyset \rightarrow G)$
"from nothing create G"

Unbind Rule: $(G \leftarrow \emptyset \rightarrow \emptyset)$
"match G and destroy it"

Transformation as Full Composition:
"first create G, then transform with p"

Rule Composition

$$D$$

$$d_1 \swarrow \qquad \searrow d_2$$

$$L_1 \leftarrow l_1 - K_1 - r_1 \rightarrow R_1 \qquad\qquad L_2 \leftarrow l_2 - K_2 - r_2 \rightarrow R_2$$
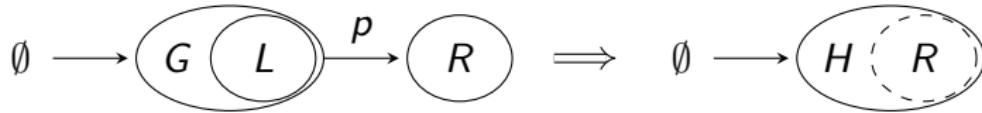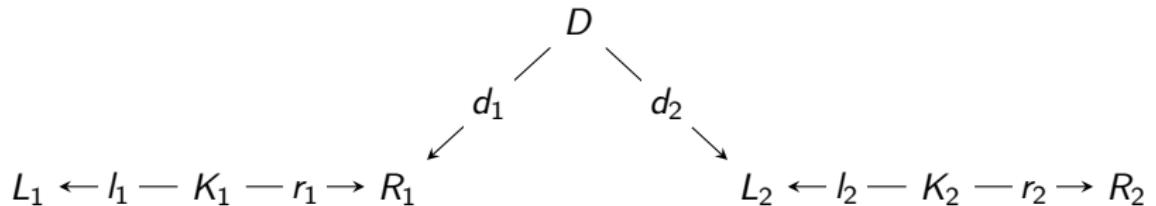
Given: two rules p_1, p_2, and the overlap $R_1 \leftarrow D \rightarrow L_2$.

Rule Composition

$$D$$

$$d_1 \qquad d_2$$

$$L_1 \leftarrow l_1 - K_1 - r_1 \rightarrow R_1 \qquad (1) \qquad L_2 \leftarrow l_2 - K_2 - r_2 \rightarrow R_2$$

$$e_1 \qquad e_2$$

$$E$$
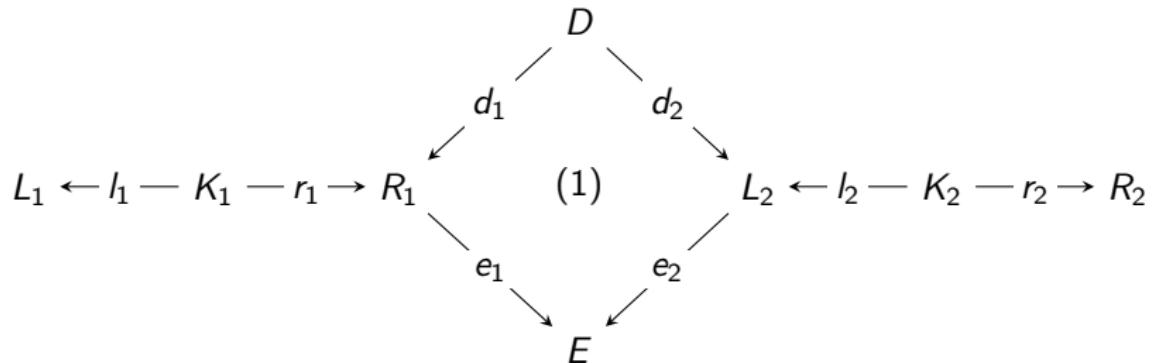
Create E as the pushout object of $R_1 \leftarrow D \rightarrow L_2$.

Rule Composition
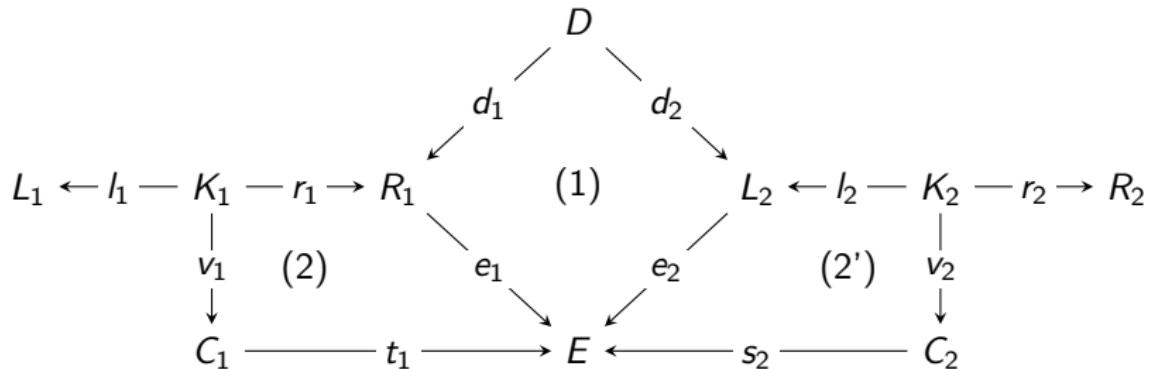
Create C_1 and C_2 as the pushout complement objects of
$K_1 \to R_1 \to E$ and $K_2 \to L_2 \to E$.

Rule Composition
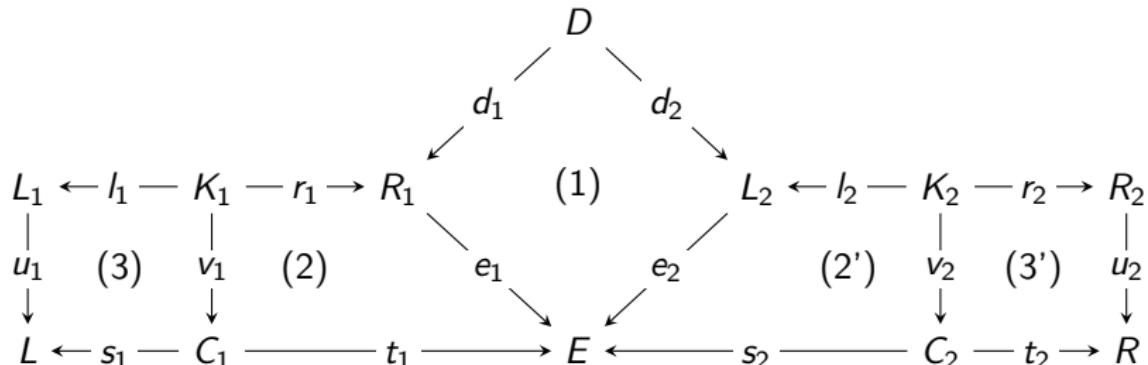
Create L and R as pushout objects of $L_1 \leftarrow K_1 \rightarrow C_1$ and $C_2 \leftarrow K_2 \rightarrow R_2$.

Rule Composition
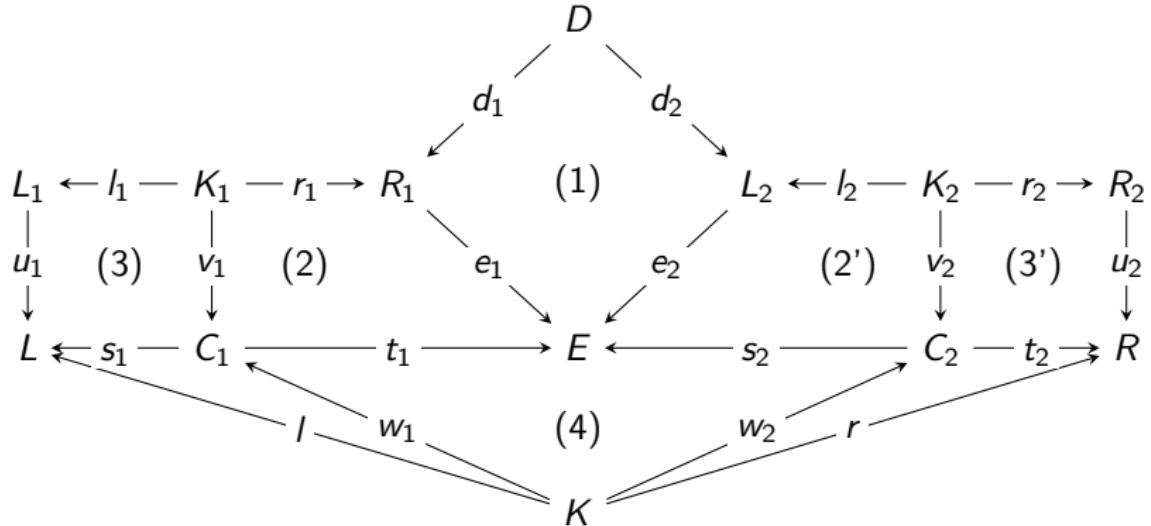
Create K as the pullback object of $C_1 \to E \leftarrow C_2$.

$$(L_1 \leftarrow K_1 \rightarrow R_1) \bullet_D (L_2 \leftarrow K_2 \rightarrow R_2) = (L \leftarrow K \rightarrow R)$$

Classes of Composition

Which $R_1 \leftarrow D \rightarrow L_2$?

Class	Pseudo Operator, $\langle op \rangle$
$D = \emptyset$	'*rcParallel*'
$D = L_2$	'*rcSuper(allowPartial=False)*'
$D = L_2', L_2 = \{L_2', L_2''\}$	'*rcSuper*'
$D = R_1$	'*rcSub(allowPartial=False)*'
$D = R_1', R_1 = \{R_1', R_1''\}$	'*rcSub*'
D unrestricted	'*rcCommon*'

Another embedded domain-specific language:

$$\langle rcExp \rangle \ :: \ \langle rcExp \rangle \ \langle op \rangle \ \langle rcExp \rangle$$
$$| \ \text{'rcBind('} \ \langle graphs \rangle \ \text{')'}$$
$$| \ \text{'rcUnbind('} \ \langle graphs \rangle \ \text{')'}$$
$$| \ \text{'rcId('} \ \langle graphs \rangle \ \text{')'}$$
$$| \ \langle rules \rangle$$

LATEX Package — Graphs
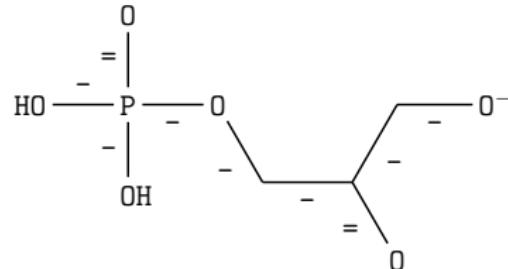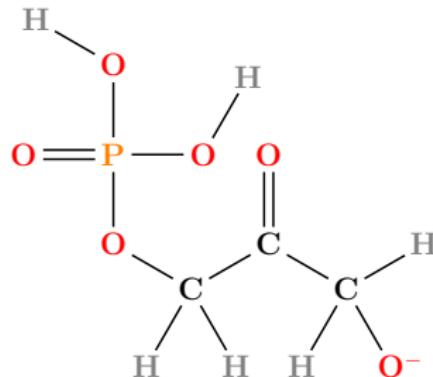
doc.tex:

```
\smiles[collapse hydrogens=false, simple carbons=false]
   {O=P(O)(O)OCC(=O)C[O-]}
\smiles[with texttt, edges as bonds=false, with colour=false]
   {O=P(O)(O)OCC(=O)C[O-]}
```

Command line:

```
pdflatex doc.tex; mod -f doc-mod.py; pdflatex doc.tex;
```

(or get a Makefile template with "`mod --get-latex`")

LATEX Package — Rules
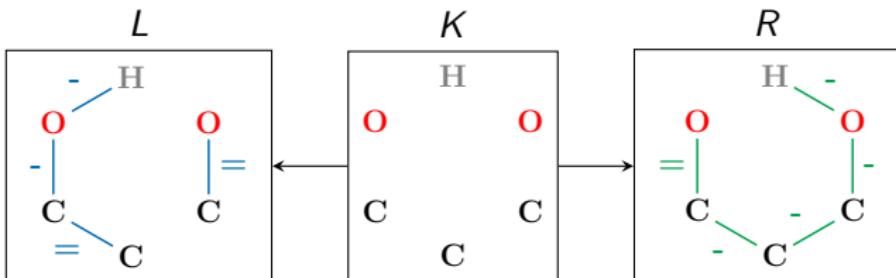
doc.tex

```
\ruleGML[edges as bonds=false]
    {data/aldol_addition_forward.gml}
    {\dpoRule[scale=0.7]}
```

Expands into:

```
\dpoRule[scale=0.7]{fileL.pdf}{fileK.pdf}{fileR.pdf}
```

Outlook

Graph Transformation:

- ► First-order terms as labels (unpublished).
- ► Attributes for local geometry (in progress).

In General:

- ► Pathways in reaction networks (submitted).
- ► Petri-net analysis for reaction networks (in progress).
- ► Tracing vertices throughout networks/pathways (in progress).

an Austrian dish

MedØl Datschgerl

"with beer"