ReCCGen
ユーザーマニュアル

第 2 版

株式会社 京都コンステラ・テクノロジーズ

目次

1　はじめに

本マニュアルでは、主に CentOS6.x(64bit)、CentOS7.x(64bit) 及び Ubuntu ディストリビューション環境における利用を想定して記述しています。また、CPU、メモリ、HDD に関しては特に制限はありませんが、なるべく高スペックのものを推奨します。

ReCGen のコンパイルに必要となるソフトウェアは以下の通りです。

ソフトウェア	推奨バージョン	URL
GCC (C / C++)	4.4.7 以降	`https://gcc.gnu.org/`
OpenBabel	2.3.2 以降	`http://openbabel.org/wiki/Main_Page`
Boost	1.41 以降	`http://www.boost.org/`
SQLite	3.5.9 以降	`https://www.sqlite.org/`
CMake	2.8.0 以降	`https://cmake.org/`
Cairo [*1]	1.8.8 以降	`https://www.cairographics.org/`
Eigen [*1]	3.2.1 以降	`http://eigen.tuxfamily.org/index.php`
libxml2 [*1]	2.7.6 以降	`http://xmlsoft.org/`
zlib [*1]	1.2.3 以降	`http://www.zlib.net/`

表 1　必須ソフトウェア一覧

上記ソフトウェアが全てインストール済みの場合は、ReCGen のコンパイルは節 2.6 （6 ページ）からの手順に従ってください。

[*1] これらは OpenBabel のコンパイルにのみ必要となるもので、ReCGen のビルドに直接必要となるものではありません。

2　インストール方法

この章では、ReCGen のコンパイルとインストールの方法を解説します。GCC 等の開発環境がインストールされていない場合は、お使いの Linux マニュアルを参照の上、予めインストールしておいて下さい。CentOS をご利用の場合は先ず以下のコマンドで EPEL リポジトリを登録して下さい。

```
$ su -    // 一旦 rootになる
# yum -y install epel-release
```

このマニュアルではコマンドのプロンプト部分が$のときは一般ユーザーで実行すべきコマンドで、#のときは root ユーザーで実行すべきコマンドである事を意味します。

2.1　CMake

CMake は Linux の種類によっては最初からインストールされていない場合がありますので、以下の手順に従ってインストールして下さい。

- CentOS の場合

```
$ su -    // 一旦 rootになる
# yum -y install cmake
```

- Ubuntu の場合

```
$ sudo apt-get install cmake -y
```

2.2　Boost

次に C++ のライブラリである Boost をインストールします。Boost も CMake と同様に以下のコマンドからインストールして下さい。

- CentOS の場合

```
# yum -y install boost boost-devel
```

- Ubuntu の場合

```
$ sudo apt-get install libboost-all-dev -y
```

バージョンの確認方法は以下の通りに行えます。

```
$ grep BOOST_LIB_VERSION /usr/include/boost/version.hpp
//  BOOST_LIB_VERSION must be defined to be the same as BOOST_VERSION
#define BOOST_LIB_VERSION "1_41"
```

この例ではバージョン 1.41 となります。

2.3 SQLite

SQLite も以下のコマンドによりインストールします。

- CentOS の場合

```
# yum -y install sqlite sqlite-devel
```

- Ubuntu の場合

```
$ sudo apt-get install sqlite3 libsqlite3-dev -y
```

バージョンの確認方法は以下の通りに行えます。

```
$ sqlite3 -version
3.11.0 2016-02-15 17:29:24 3d862f207e3adc00f78066799ac5a8c282430a5f
```

この例ではバージョン 3.11.0 となります。

2.4 OpenBabel 用ライブラリ

OpenBabel のビルドに必要となる以下のライブラリをインストールします。

Cairo　ベクトル画像を描画するライブラリーです。
Eigen　線形代数の C++ テンプレートライブラリです。
libxml2　XML を読み書きするためのライブラリです。
zlib　データの圧縮および伸張を行うためのライブラリです。

以下のコマンドでインストールします。

- CentOS の場合

```
# yum -y install eigen3-devel libxml2 libxml2-devel
# yum -y install cairo cairo-devel zlib zlib-devel
```

- Ubuntu の場合

```
$ sudo apt install libeigen3-dev libxml2-dev -y
$ sudo apt install libcairo2-dev zlib1g-dev -y
```

Cairo のバージョン確認は以下のように行なえます。この場合はバージョン 1.8.8 となります。

```
$ cat /usr/include/cairo/cairo-version.h
#ifndef CAIRO_VERSION_H
#define CAIRO_VERSION_H

#define CAIRO_VERSION_MAJOR 1
#define CAIRO_VERSION_MINOR 8
#define CAIRO_VERSION_MICRO 8

#endif
```

Eigen のバージョン確認は以下のように行なえます。この場合はバージョン 3.2.5 となります。

```
$ grep -e VERSION /usr/include/eigen3/Eigen/src/Core/util/Macros.h | head -n 3
#define EIGEN_WORLD_VERSION 3
#define EIGEN_MAJOR_VERSION 2
#define EIGEN_MINOR_VERSION 5
```

libxml2 のバージョン確認は以下のように行なえます。この場合はバージョン 2.7.6 となります。

```
$ grep LIBXML_DOTTED_VERSION /usr/include/libxml2/libxml/xmlversion.h
 * LIBXML_DOTTED_VERSION:
#define LIBXML_DOTTED_VERSION "2.7.6"
```

zlib のバージョン確認は以下のように行なえます。この場合はバージョン 1.2.3 となります。

```
$ grep -e "^#.*ZLIB_VERSION" /usr/include/zlib.h
#define ZLIB_VERSION "1.2.3"
```

2.5 OpenBabel のインストール

Boost ライブラリや Eigen 等の必要ライブラリのインストールが出来れば OpenBabel をコンパイルするための条件が整います。以下の手順に従って OpenBabel をインストールして下さい。

① アーカイブの展開

適当なディレクトリにおいて、ダウンロードした”openbabel-2.3.2.tar.gz”を展開して下さい。

```
$ tar xvfz openbabel-2.3.2.tar.gz    // アーカイブの展開
openbabel-2.3.2/
openbabel-2.3.2/AUTHORS
openbabel-2.3.2/ChangeLog
openbabel-2.3.2/ChangeLog.1.x
……
```

② ビルド用のディレクトリの作成

ビルド用のディレクトリを作成して下さい。ここでは名前を”build”とします。

```
$ cd openbabel-2.3.2
$ mkdir build     // ディレクトリ作成
$ cd build        // 作成したディレクトリに移動
```

③ CMake の実行

以下の様に cmake を実行して下さい。デフォルトの設定では/usr/local の下にインストールされますが、指定する場合は “-DCMAKE_INSTALL_PREFIX=(path to install)” というオプションを追加して下さい。

```
$ cmake ..
```

④ コンパイルとインストール

以下の様に make と make install を実行して下さい。

```
$ make
$ su
# make install
```

インストールとバージョンの確認は以下の通りです。

```
$ babel -V
Open Babel 2.3.2 -- Aug  8 2014 -- 20:53:08
```

2.6 ReCGen のインストール

Cmake を利用してコンパイル・インストールするので、手順は前述の OpenBabel と概ね同じです。

① アーカイブの展開

```
$ tar xvfz tar xvfz ReCGen-1.0.tgz     // アーカイブの展開
ReCGen/
ReCGen/src/
ReCGen/src/Fragmentation.cpp
ReCGen/src/Build.cpp
……
```

展開された後のディレクトリ構成は以下のようになります。

ReCGen
- DB　サンプルのフラグメント DB が入っています。
- doc　マニュアル等のドキュメントが入っています。
- examples　チュートリアル用の入力ファイルが入っています。
- src　ソースコードが入っています。

② ビルド用のディレクトリの作成

```
$ cd ReCGen
$ mkdir build     // ディレクトリ作成
$ cd build        // 作成したディレクトリに移動
```

③ CMake の実行

OpenBabel が/usr/local や/usr/lib の下インストールされている場合は以下の様に cmake を実行して下さい。ホームディレクトリの下など/usr/local 以外の場合 “-DOpenBabel_DIR=(path to babel)” オプションを追加して下さい。

```
// /usr/localの下にOpenBabelがある場合
$ cmake ..

// OpenBabelのディレクトリを明示する場合
$ cmake -DopenBabel_DIR=$HOME/appl ..
```

/usr/local/bin 以外のインストールディレクトリを指定する場合は以下の様に cmake を実行して下さい。以下の例はホームディレクトリの appl/bin の下にインストールします。

```
$ cmake -DCMAKE_INSTALL_PREFIX=$HOME/appl ..
```

④ コンパイルとインストール

```
$ make
$ su
# make install
```

3　概要

本プログラムでは、フラグメント作成ツール、DB マージツール、構造生成ツールの 3 つの機能を提供しております。図 1 にそれらのツールの関係を示します。計算の流れとしては先ずフラグメント作成ツールを使いフラグメント DB を作成して、次に構造生成ツールに母核構造とフラグメント DB を入力して新規構造を生成します。生成される化合物数は、入力するフラグメント DB のフラグメント数の大きさに比例します。DB マージツールは別々の化合物ライグラリから作成したフラグメント DB を一つの DB にまとめて、より大きなフラグメント DB を作成します。

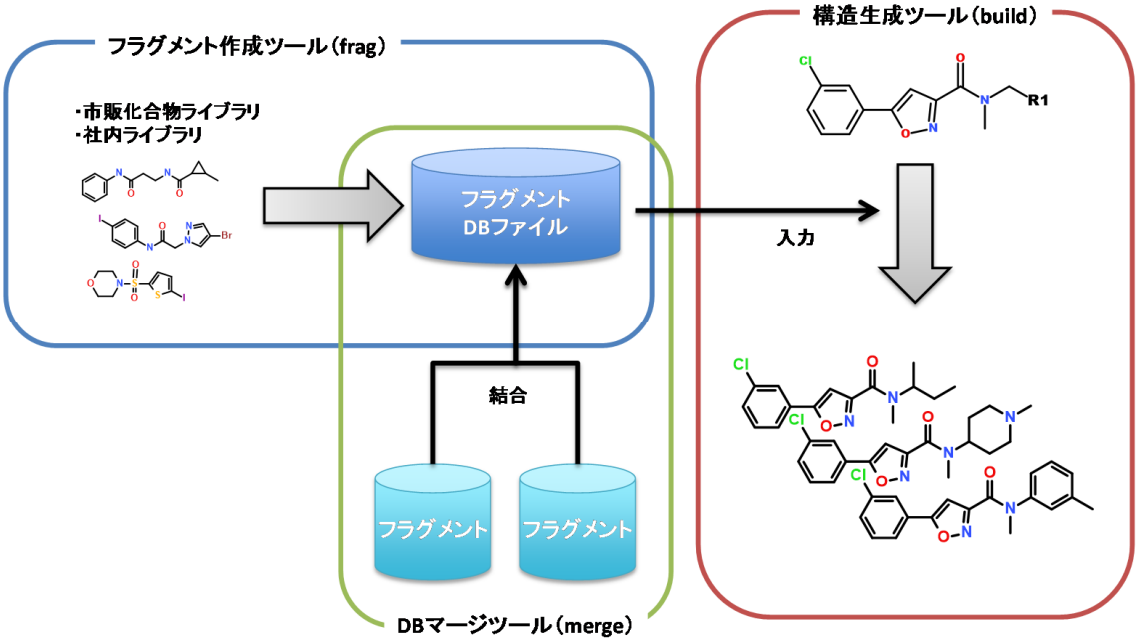

図 1　フラグメント作成から構造生成までの全体像

4　フラグメント DB の作成

4.1　SD ファイルからフラグメント DB を作成する

入力の SD ファイル（input.sdf とします）からフラグメント DB（output.db とします）を生成するには以下の様にコマンドを実行します。

```
$ frag input.sdf output.db
```

上記の例では ECFP2〜ECFP8 までのフラグメントを生成します。ECFP8 より大きなフラグメントも生成する場合は-d オプションを使用して下さい。このオプションはフラグメントの大きさの上限を指定します。以下の例は ECFP10 までのフラグメントを作成します。

```
$ frag -d 10 input.sdf output.db
```

一方、フラグメントの大きさの下限を指定する場合は-m オプションを使用します。以下の例では ECFP6 のみのフラグメントを作成します。

```
$ frag -d 6 -m 6 input.sdf output.db
```

4.2　2 つの DB をマージする

2 つのフラグメント DB をマージして 1 つのフラグメント DB にまとめるには以下のコマンドを実行して下さい。

```
$ merge A.db B.db C.db
```

この例では A.db と B.db という 2 つの DB をマージして C.db を作成します。このとき重複するフラグメントは取り除かれます。DB をマージするツールは主に大量の化合物ライブラリを複数のファイルに分割してフラグメント作成を行う場合に、最後の集計として利用できます。

5　構造の生成

ReCGen のアーカイブを展開してできる examples ディレクトリに、サンプルの入力 MOL ファイルが入っています。また、DB ディレクトリにはサンプルのフラグメント DB（DrugBank_M.db）が入っています。ここで解説する例は、これらのファイルを利用して試すことができます。以下の表 2, 3 にサンプル入力ファイルを使用した計算結果と、その計算を行った計算環境を示します。

MOL ファイル	説明	計算時間	生成化合物数
sample1.mol	1 箇所変換	00:00:11	1118
sample2.mol	2 箇所変換	00:18:06	80784
sample3.mol	2 段階変換	05:34:47	666951
sample4.mol	1 段階補間	00:00:01	33
sample5.mol	2 段階補間	00:28:27	15210

表 2　サンプル計算のまとめ

CPU	AMD Phenom(tm) II X6 1055T
メモリ	8GB DDR3-1333
HDD	1.5TB SAMSUNG HD154UI
OS	Ubuntu 16.04.3 LTS

表 3　サンプル計算を実行した計算環境

5.1　基本的な構造生成

先ず構造式描画ツールを使って、入力する構造式を描画します。描画ツールはどの様なものでも構いませんが、本マニュアルの例では JChemPaint（URL: `https://jchempaint.github.io`）を使用します。入力構造は描画ツールにて母核構造と構造置換する位置を R1, R2 等の R グループを書いて下さい。例えばベンゼン環上に一カ所置換する場合は図 2 の様になります。入力構造が描画出来たらMDL　MOL形式でファイルに保存して下さい。なお、他の構造描画ツールで R1 や R2 を記述することが出来ないものもあります。そのようなときは、R1 → Rf、R2 → Db、R3 → Sg、R4 → Bh と R グループを重元素で置き換えて記述して下さい。

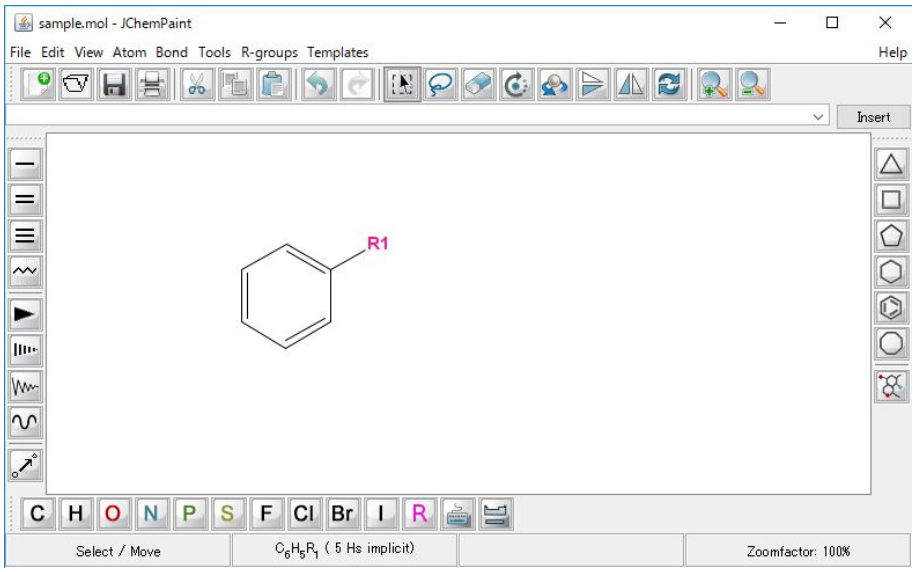

図 2　1 カ所変換の描画例

上記の例で描画ツールから出力される MOL ファイルの先頭部分は以下の様になります。

```
CDK     1003161336

7  7  0  0  0  0  0  0  0  0999 V2000
 -4.6333    1.6833    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
 -3.3343    0.9333    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
 -3.3343   -0.5667    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
 -4.6333   -1.3167    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
 -5.9324   -0.5667    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
 -5.9324    0.9333    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
 -2.0353    1.6833    0.0000 R#  0  0  0  0  0  0  0  0  0  0  0  0
1  2  2  0  0  0  0
2  3  1  0  0  0  0
```

大抵の描画ツールでは、出力されたばかりの状態では先頭行が空行のままです。必須ではありませんが、なるべくこの部位に母核名などの名前を記述して下さい。生成された構造の分子名の一部として利用されます。

この例の入力ファイルは “sample1.mol” というファイル名で examples ディレクトリに入っています。これを利用して以下のコマンドでフラグメント DB から構造を生成できます。

```
$ build DrugBank_M.db sample1.mol output1.sdf
```

生成された構造は output1.sdf に出力されます。デフォルトの設定では崩れた2次元構造が出力されます。出力構造を整形する場合は-2 オプションを付けて下さい。

```
$ build -2 DrugBank_M.db sample1.mol output1.sdf
```

5.2　複数個所の構造生成

複数個所の R グループに対する構造変換も行えます。R1 と R2 の 2 カ所の変換を行う場合は以下の図 3 のような入力構造を描いて下さい。その後は前節の同様に、MOL 形式でファイル出力して build コマンドを使用して下さい。利用できる R グループの数は R1–8 までです。しかしながら R グループの数が増えると飛躍的にメモリを消費し、計算時間が延びるため R1–3 程度を推奨します。図 3 の入力例では、弊社計算環境で 18 分程度の時間がかかります。この例の入力ファイルは “sample2.mol” というファイル名で examples ディレクトリに入っています。

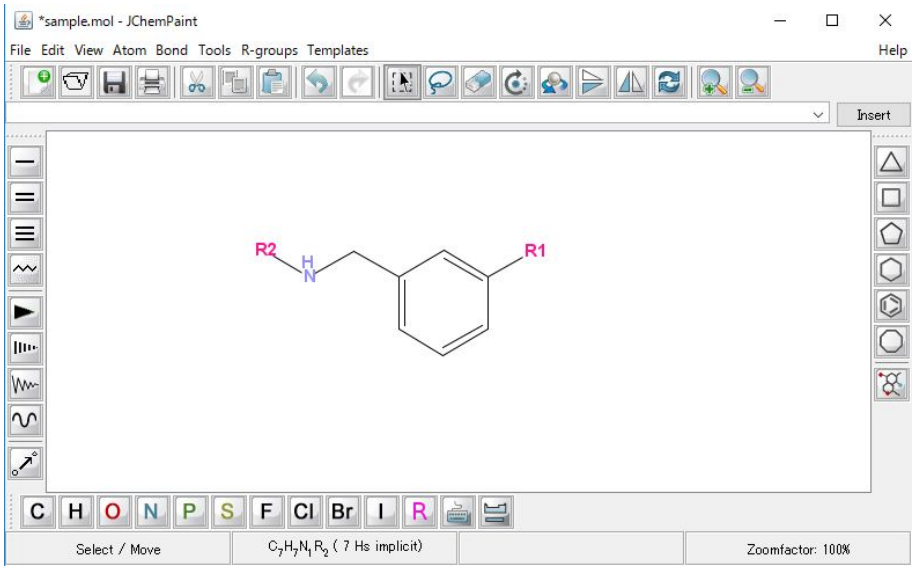

図 3　2 カ所変換の描画例

5.3　多段階の生成

前節までは 1 カ所につき 1 段階の構造置換を行う例を解説しました。フラグメント DB が余程巨大なものでない限り 1 段階で生成出来る構造は精々数万個程度です。1 カ所あたりの置換構造のバリエーションを増やすためには構造を多段階生成する必要があります。2 段階で構造生成するためには、図 4 の様に R グループ同士を直接結合させた入力構造を記述して下さい。この例では 2 段階で構造を生成します。入力ファイルは “sample3.mol” というファイル名で examples ディレクトリに入っていますが、実行には非常に時間がかかります。弊社計算環境でサンプルのフラグメント DB を使用した場合に 5 時間半程度です。

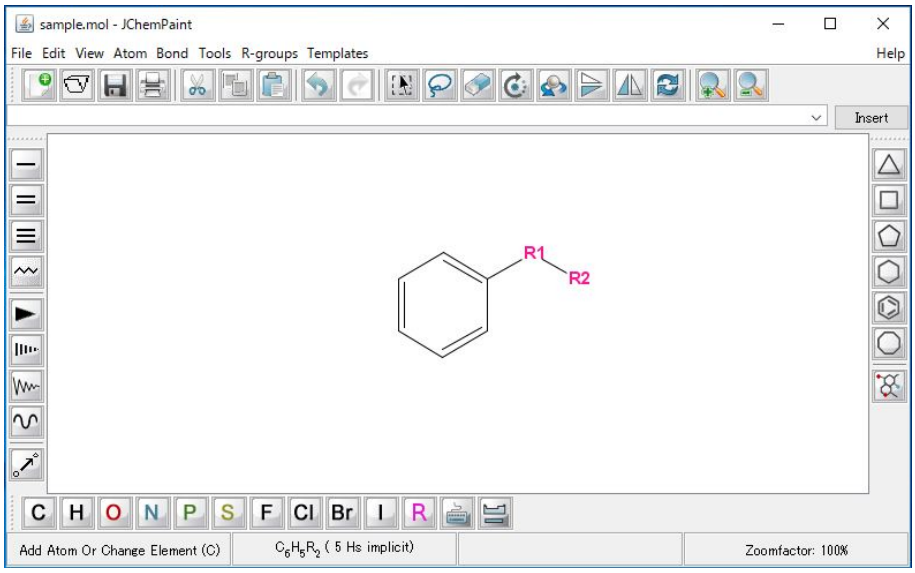

図 4　2 段階変換の描画例

5.4　リンカー構造の生成（補間）

リンカー構造を生成する場合は図 5 のような入力構造を描画します。図 5 の左側が 1 段階で構造生成する場合で、右側が 2 段階で構造生成する場合となります。リンカーを生成する場合は、1 段のみでは生成されるバリエーションが少ないので 2 段階以上の設定で構造生成することを強く推奨します。examples ディレクトリには “sample4.mol”（1 段階）、“sample5.mol”（2 段階）というファイル名で入っています。サンプルのフラグメント DB を使用した場合に生成される化合物数は、1 段階の方は 33 個で 2 段階の方は 15210 個です。

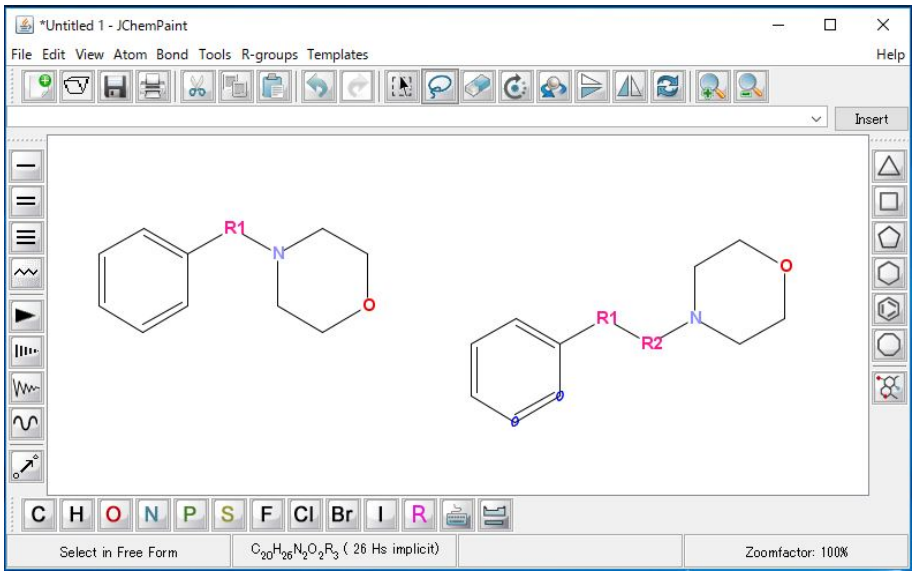

図 5　2 構造間を補間するリンカー構造を生成する描画例

5.5　その他

今までは何らかの母核構造がある場合の構造生成について解説しましたが、下図のような入力構造を描画することで、母核構造無しで構造生成する事も可能です。ここで、お使いの描画ツールによっては図 6 のような特殊な構造を描画出来ない場合があります。そのような場合は前述した通り、R1 → Rf、R2 → Db と置き換えて描画して下さい。

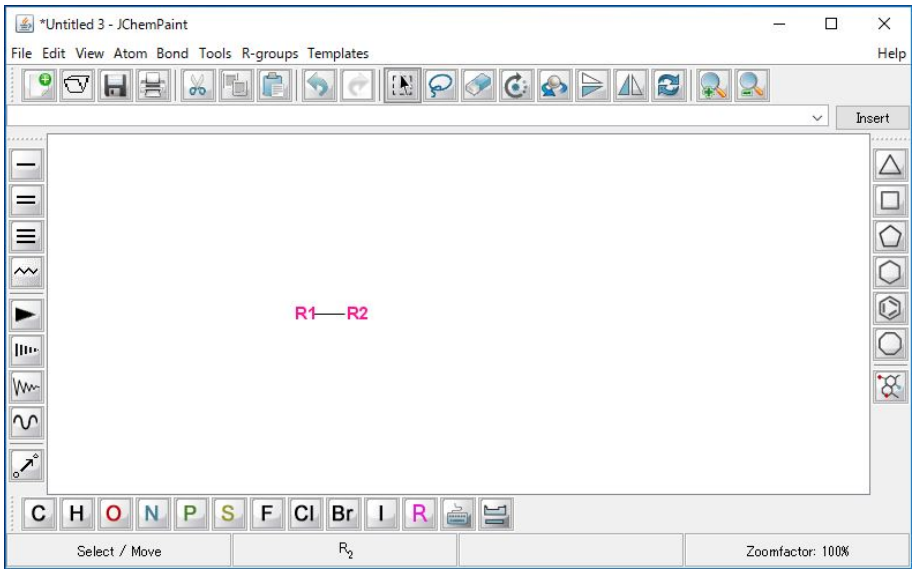

図 6　母核構造が無い場合の描画例