```
SetDirectory["C:\\drorbn\\AcademicPensieve\\2014-01"];
Make[targets_List, rules_List] := Module[{},
  alltargets = targets;

 ]

Make[
 {"AbstractionChallenge.png",
  "AbstractionChallenge-1.png", "AbstractionChallenge-2.png"},
 {
  {"AbstractionChallenge.png", "AbstractionChallenge-1.png",
     "AbstractionChallenge-2.png"} /; "nb/AbstractionChallenge.pdf" :> (
    Read["!convert -density 200 -scene 1 -background white -trim
       nb/AbstractionChallenge.pdf AbstractionChallenge.png"];
    imgs = Import /@ {"AbstractionChallenge-1.png",
       "AbstractionChallenge-2.png"};
    DeleteFile /@ {"AbstractionChallenge.png", "AbstractionChallenge-1.png",
      "AbstractionChallenge-2.png"};
    imgs = ImageTake[#, {20, -200}] & /@ imgs;
    imgs = ImageCrop /@ imgs;
    imgs = RemoveAlphaChannel[#, White] & /@ imgs;
    dims = Max /@ Transpose[ImageDimensions /@ imgs];
img = ImageAssemble[imgs /. x_Image :> ImageCrop[x, dims, Padding -> Automatic]];
    img = ImageCrop[img];
    {
     Export["AbstractionChallenge.png", img],
     Export["AbstractionChallenge-1.png", imgs[[1]]],
     Export["AbstractionChallenge-2.png", imgs[[2]]]
    }
   )
 }
]
```

```
{AbstractionChallenge.png, AbstractionChallenge-1.png, AbstractionChallenge-2.png}
```

```
rules = {
  {"AbstractionChallenge.png", "AbstractionChallenge-1.png",
     "AbstractionChallenge-2.png"} /; "nb/AbstractionChallenge.pdf" :> (
    Read["!convert -density 200 -scene 1 -background white -trim
       nb/AbstractionChallenge.pdf AbstractionChallenge.png"];
    imgs = Import /@ {"AbstractionChallenge-1.png",
       "AbstractionChallenge-2.png"};
    DeleteFile /@ {"AbstractionChallenge.png", "AbstractionChallenge-1.png",
      "AbstractionChallenge-2.png"};
    imgs = ImageTake[#, {20, -200}] & /@ imgs;
    imgs = ImageCrop /@ imgs;
    imgs = RemoveAlphaChannel[#, White] & /@ imgs;
    dims = Max /@ Transpose[ImageDimensions /@ imgs];
img = ImageAssemble[imgs /. x_Image :> ImageCrop[x, dims, Padding -> Automatic]];
    img = ImageCrop[img];
    {
     Export["AbstractionChallenge.png", img],
     Export["AbstractionChallenge-1.png", imgs[[1]]],
     Export["AbstractionChallenge-2.png", imgs[[2]]]
    }
   )
 }
```

```
{{AbstractionChallenge.png, AbstractionChallenge-1.png,
    AbstractionChallenge-2.png} /; nb/AbstractionChallenge.pdf :>
  (Read[!convert -density 200 -scene 1 -background white -trim
      nb/AbstractionChallenge.pdf AbstractionChallenge.png];
   imgs = Import /@ {AbstractionChallenge-1.png, AbstractionChallenge-2.png};
   DeleteFile /@ {AbstractionChallenge.png,
     AbstractionChallenge-1.png, AbstractionChallenge-2.png};
   imgs = (ImageTake[#1, {20, -200}] &) /@ imgs; imgs = ImageCrop /@ imgs;
   imgs = (RemoveAlphaChannel[#1, White] &) /@ imgs;
   dims = Max /@ Transpose[ImageDimensions /@ imgs];
   img = ImageAssemble[imgs /. x_Image :> ImageCrop[x, dims, Padding -> Automatic]];
   img = ImageCrop[img]; {Export[AbstractionChallenge.png, img],
    Export[AbstractionChallenge-1.png, imgs[[1]]],
    Export[AbstractionChallenge-2.png, imgs[[2]]]})}
```

```
Replace[rules, a_RuleDelayed :> List @@ a, {1}]
```

Import::nffil : File not found during Import. ≫

Import::nffil : File not found during Import. ≫

DeleteFile::nffil : File not found during DeleteFile[AbstractionChallenge−1.png]. ≫

DeleteFile::nffil : File not found during DeleteFile[AbstractionChallenge−2.png]. ≫

ImageTake::imginv : Expecting an image or graphics instead of $Failed. ≫

ImageTake::imginv : Expecting an image or graphics instead of $Failed. ≫

ImageCrop::imginv : Expecting an image or graphics instead of ImageTake[$Failed, {20, −200}]. ≫

ImageCrop::imginv : Expecting an image or graphics instead of ImageTake[$Failed, {20, −200}]. ≫

RemoveAlphaChannel::imginv : Expecting an image or graphics instead of ImageCrop[ImageTake[$Failed, {20, −200}]]. ≫

RemoveAlphaChannel::imginv : Expecting an image or graphics instead of ImageCrop[ImageTake[$Failed, {20, −200}]]. ≫

RemoveAlphaChannel::imginv : Expecting an image or graphics instead of ImageCrop[ImageTake[$Failed, {20, −200}]]. ≫

General::stop : Further output of RemoveAlphaChannel::imginv will be suppressed during this calculation. ≫

ImageDimensions::imginv :
Expecting an image or graphics instead of RemoveAlphaChannel[ImageCrop[ImageTake[$Failed, {20, −200}]], GrayLevel[1]]. ≫

ImageDimensions::imginv :
Expecting an image or graphics instead of RemoveAlphaChannel[ImageCrop[ImageTake[$Failed, {20, −200}]], GrayLevel[1]]. ≫

Transpose::nmtx : The first two levels of the one−dimensional list
{ImageDimensions[RemoveAlphaChannel[ImageCrop[ImageTake[$Failed, {20, −200}]], GrayLevel[1]]], ImageDimensions[
RemoveAlphaChannel[ImageCrop[ImageTake[$Failed, {20, −200}]], GrayLevel[1]]]}
cannot be transposed. ≫

ImageAssemble::imginv :
Expecting an image or graphics instead of RemoveAlphaChannel[ImageCrop[ImageTake[$Failed, {20, −200}]], GrayLevel[1]]. ≫

ImageCrop::imginv : Expecting an image or graphics instead of
ImageAssemble[{RemoveAlphaChannel[ImageCrop[ImageTake[$Failed, {20, −200}]], GrayLevel[1]], RemoveAlphaChannel[
ImageCrop[ImageTake[$Failed, {20, −200}]], GrayLevel[1]]}]. ≫

General::stop : Further output of ImageCrop::imginv will be suppressed during this calculation. ≫

```
{{{AbstractionChallenge.png, AbstractionChallenge-1.png,
     AbstractionChallenge-2.png} /; nb/AbstractionChallenge.pdf,
   {AbstractionChallenge.png, AbstractionChallenge-1.png,
    AbstractionChallenge-2.png}}}
```

```
rules[[1]] // FullForm
```

```
RuleDelayed[
 Condition[List["AbstractionChallenge.png", "AbstractionChallenge-1.png",
   "AbstractionChallenge-2.png"], "nb/AbstractionChallenge.pdf"],
 CompoundExpression[Read["!convert -density 200 -scene 1 -background white
     -trim nb/AbstractionChallenge.pdf AbstractionChallenge.png"], Set[imgs,
   Map[Import, List["AbstractionChallenge-1.png", "AbstractionChallenge-2.png"]]],
  Map[DeleteFile, List["AbstractionChallenge.png",
    "AbstractionChallenge-1.png", "AbstractionChallenge-2.png"]],
  Set[imgs, Map[Function[ImageTake[Slot[1], List[20, -200]]], imgs]],
  Set[imgs, Map[ImageCrop, imgs]],
  Set[imgs, Map[Function[RemoveAlphaChannel[Slot[1], White]], imgs]],
  Set[dims, Map[Max, Transpose[Map[ImageDimensions, imgs]]]],
  Set[img, ImageAssemble[ReplaceAll[imgs, RuleDelayed[
      Pattern[x, Blank[Image]], ImageCrop[x, dims, Rule[Padding, Automatic]]]]]],
  Set[img, ImageCrop[img]], List[Export["AbstractionChallenge.png", img],
   Export["AbstractionChallenge-1.png", Part[imgs, 1]],
   Export["AbstractionChallenge-2.png", Part[imgs, 2]]]]]
```