```
# Screening Examination of Premature Infants for Retinopathy of Prematurity
# Release Date: 2006
# Guideline Developer: Section on Ophthalmology, American Academy of Pediatrics,
# American Academy of Ophthalmology and American Association for Pediatric Ophthalmology
and Strabismus

# Assumes only one patient is in working memory at a time

#created on: Nov 11, 2010
package edu.chop.cbmi.cds.ROP

#list any import classes here.
import java.util.ArrayList;
import edu.chop.cbmi.cds.Patient;
import edu.chop.cbmi.cds.Patient.*;
import edu.chop.cbmi.cds.Localization;
import edu.chop.cbmi.cds.DocFact;
import edu.chop.cbmi.cds.DocFact.*;
import edu.chop.cbmi.cds.FuzzyDate;
import edu.chop.cbmi.cds.Regex;

import org.joda.time.*;
import org.joda.time.format.*;
import java.util.regex.Pattern;
import java.util.regex.Matcher;

#declare any global variables here

# ROP fact dictionary
# Describe patients who are not eligible for ROP screening
declare ROPExclude
    reason : String
end

# Describe candidate reasons why a patient may or may not be eligible for ROP screening.
declare ROPEligibleCandidate
    reason : String
end

# Describe the final reason why a patient is or is not eligible for ROP screening.
declare ROPEligibleFinal
  eligible : Boolean
  reason : String
end

# Describe patients who had an unstable clinical course
declare ROPUnstableCourse
    reason : String
end

# Identify patients who are believed to be at high risk
declare ROPHighRisk
    reason : String
end
```

```
# Describe imperatives related to ROP that may apply
declare ROPImperative
    directive : String
end

# Describe all findings reported on a single day
declare ROPFinding
  date : DateMidnight
  stage : Integer
  zone : Integer
  immatureVascularization : Boolean
  completeVascularization : Boolean
  plusDisease : Boolean
  resolved: Boolean       # indicates whether this was marked as resolved in the chart
  description : String    # plain text description of this finding
  url : String            # URL for the location in the EHR to find the documentation
  source : String         # describe best source where documentation was found, may be
                          # "Diagnosis", "Problem", "MedicalHistory", "BirthHistory", or
"Narrative"
                          # in that order of priority
end

# identify descriptive fragments that may contain retinopathy descriptions
declare ROPDescription
  source    : String   # describe source where documentation was found, may be
                       # "Diagnosis", "Problem", "MedicalHistory", "BirthHistory", or
"Narrative"
  date : DateMidnight  # best estimate for this observation
  encounter : Encounter
  text : String
end

# additional annotations that can be attached to findings as part of follow-up assessment
declare ROPRegressing
  finding : ROPFinding
  previous : ROPFinding
end

declare ROPWorsening
  finding : ROPFinding
  previous : ROPFinding
end

# Describe schedule for screening
declare ROPFollowupCandidate
  finding : ROPFinding
  interval : Interval
  reason : String
end

# Schedule
declare ROPSchedule
  type : String        # may be "first-screening" "followup-screening"  "conclude-
```

```
screening" or "ablation"
  priority : Integer   # priority of this schedule recommendation, lower number have
higher priority (e.g. ablation is higher priority than follow-up)
  finding : ROPFinding
  interval : Interval
  reason : String
end

# last valid observation
declare ROPLastValid
    newborn : Boolean        # indicate if this was completed in the newborn/NICU period
    last : ROPFinding        # last ROP result observed
end

# report final ROP recommendations to calling application.  Describe the final eligibility
# determination, any imperatives that may apply, interpretation of timeliness and
signficance
# of prior retinopathy findings, and schedule for future screening, ablation, and
conlusion
# of screening
declare ROPRecommendation
    eligible : ROPEligibleFinal
    imperatives : ArrayList
    findings : ArrayList
    schedule : ArrayList
    summaryLast : String                    # describe the most recent screening
    summaryFollowup : String                # describe indicated follow-up (if any)
    summaryAction : String                  # describe the next steps, or action to take
(if any)
    summaryStatement : String               # summary statement for display on summary
face page
    summaryBriefStatement : String          # abbreviated summary statement for display
on summary face page
    flag : Boolean                          # indicate whether this row should be
flagged
    actionProblemList : String              # describe documentation that should be
added to problem list
    actionBirthHx : String                  # describe documentation that should be
added to birth history
    actionPastMedicalHx : String            # describe documentation that should be
added to past medical history
    resourceUrl : String                    # provide URL for preemie smart set if
needed
    resultUrl : String                      # provide URL for most recent result if
available
end

#Exclusion Criterion: Term infants
rule "exclude term infants"
    ruleflow-group "rop-risk-eligibility"
    when
        # identify patients with gestational age >= 35 0/7 weeks
                $p: Patient($gestAge: gestAge >= 35)
        then
```

```
                # exclude full term patients from retinal screening exams
                ROPExclude fact = new ROPExclude();
                #fact.setPatient($p);
                fact.setReason("gestational age " + $gestAge + " weeks");
                insert(fact);
end

# RECOMMENDATION: 1. Candidates for retinal screening exams
# Eligibility criteria may overlap.  As final step "roll up" eligibility criteria
# and exclusions to determine final eligibility.  Three non-overlapping possibilities
# exist: (1) eligible and not excluded; (2) excluded; (3) not eligible and not excluded
rule "eligible and not excluded"
    ruleflow-group "rop-final-eligibility"
    when
        # Iterate over patients
        $p: Patient()
        # Check to make sure patient is not excluded
        not (exists (ROPExclude()))
        # Find at least one eligibility reason for this patient
        $eligibleList: ArrayList(size > 0) from collect (ROPEligibleCandidate())
    then
        # Indicate reason(s) why patient is eligible
                ROPEligibleFinal fact = new ROPEligibleFinal();
                #fact.setPatient($p);
                fact.setEligible(true);
                # loop over all candidate reasons why patient may be eligible and
aggregate
                # into a single explanation
                String eligibleReason = "";
                for(ROPEligibleCandidate eligible : (ArrayList<ROPEligibleCandidate>)
$eligibleList) {
                  eligibleReason = eligibleReason + (eligibleReason.length() > 0 ? ", " :
"") + eligible.getReason();
                }
                fact.setReason(eligibleReason);
                insert(fact);
                # publish eligibility
                insert(new DocFact(DocFact.ROP_ELIGIBLE_ID, eligibleReason));
end

rule "excluded not eligible"
    ruleflow-group "rop-final-eligibility"
    when
        # Iterate over patients
        $p: Patient()
        # Find at least one exclusion reason for this patient
        $excludeList: ArrayList(size > 0) from collect (ROPExclude())
    then
        # Indicate reason(s) why patient is NOT eligible
                ROPEligibleFinal fact = new ROPEligibleFinal();
                #fact.setPatient($p);
                fact.setEligible(false);
                # loop over all reasons why patient is excluded and aggregate
                # into a single explanation
```

```
                String excludeReason = "";
                for(ROPExclude exclude :
                  (ArrayList<ROPExclude>)$excludeList) excludeReason = excludeReason + ",
" + exclude.getReason();
                fact.setReason(excludeReason);
                insert(fact);
end

rule "not excluded not eligible"
    ruleflow-group "rop-final-eligibility"
    when
        # Iterate over patients
        $p: Patient()
        # skip patients who have an eligibility reason
        not (exists (ROPEligibleCandidate()))
        # skip patients who have an exclusion reason
        not (exists (ROPExclude()))
    then
        # Indicate reason why patient is NOT eligible
                ROPEligibleFinal fact = new ROPEligibleFinal();
                #fact.setPatient($p);
                fact.setEligible(false);
                fact.setReason("This patient does not meet eligibility criteria for ROP
screening");
                insert(fact);
end

# Conditional:  1.1 Infants with a birth weight of less than 1500 g or gestational age of
# 30 weeks or less (as defined by the attending neonatologist)
rule "BW less than 1500 g"
    ruleflow-group "rop-risk-eligibility"
        when
            # identify patients with birth weight <= 1.5 kg
                $p: Patient($birthWeightKG: birthWeightKG <= 1.5)
        then
            # eligible candidate for retinal screening exams based on birth weight
                ROPEligibleCandidate fact = new ROPEligibleCandidate();
                #fact.setPatient($p);
                fact.setReason("birth weight " + $birthWeightKG*1000 + " grams");
                insert(fact);
end

rule "gest age 30 weeks or less"
    ruleflow-group "rop-risk-eligibility"
        when
            # identify patients with gestational age <= 30 6/7 weeks
                $p: Patient($gestAge: gestAge < 31)
        then
            # eligible candidate for retinal screening exams based on gestational age
                ROPEligibleCandidate fact = new ROPEligibleCandidate();
                #fact.setPatient($p);
                fact.setReason("gestational age " + $gestAge + " weeks");
                insert(fact);
end
```

```
# Conditional: 1.2 Birth weight between 1500 and 2000 g or gestational age of more than 30
weeks
# with an unstable clinical course, including those requiring cardiorespiratory support
and who
# are believed by their attending pediatrician or neonatologist to be at high risk, should
have
# retinal screening examinations performed after pupillary dilation using binocular
indirect
# ophthalmoscopy to detect ROP.
rule "BW 1500 to 2000 g with unstable clinical course"
    ruleflow-group "rop-risk-eligibility"
        when
                # identify patients with birth weight between 1.5 and 2.0 kg
                $p: Patient($birthWeightKG: birthWeightKG > 1.5 && birthWeightKG <= 2.0)
                # and unstable clinical course
            $unstable: ROPUnstableCourse()
        then
            # eligible candidate for retinal screening exams based on birth weight and
unstable clinical course
                ROPEligibleCandidate fact = new ROPEligibleCandidate();
                #fact.setPatient($p);
                fact.setReason("birth weight " + $birthWeightKG*1000 + " grams, and
unstable clinical course: "
                                + $unstable.getReason());
                insert(fact);
end

rule "gest age more than 30 weeks with unstable clinical course"
    ruleflow-group "rop-risk-eligibility"
        when
                # identify patients with gestational age more than 30 6/7 weeks
                $p: Patient($gestAge: gestAge >= 31)
                # and unstable clinical course
            $unstable: ROPUnstableCourse()
        then
            # eligible candidate for retinal screening exams based on ggestational age and
unstable clinical course
                ROPEligibleCandidate fact = new ROPEligibleCandidate();
                #fact.setPatient($p);
                fact.setReason("gestational age " + $gestAge + " weeks, and unstable
clinical course: "
                                + $unstable.getReason());
                insert(fact);
end

# Decision Variable: Unstable clinical course (1)
# unstable clinical course, including those requiring cardiorespiratory support
rule "unstable course cardiorespiratory support"
    ruleflow-group "rop-risk-eligibility"
        when
            $p: Patient()
            # determine whether patient required cardiorespiratory support based on
diagnoses
```

```
            # TODO: identify diagnoses that indicate cardiorespiratory support
            exists (Diagnosis(icd9 matches "TBD\\.TBD.*") from $p.getDiagnoses())
        then
            ROPUnstableCourse fact = new ROPUnstableCourse();
            #fact.setPatient($p);
            fact.setReason("required cardiorespiratory support");
            insert(fact);
end

# Decision Variable: Unstable clinical course (2)
# unstable clinical course, including those who are believed by their attending
# pediatrician or neonatologist to be at high risk
rule "unstable course high risk"
    ruleflow-group "rop-risk-eligibility"
        when
            ROPHighRisk($reason: reason)
        then
                ROPUnstableCourse fact = new ROPUnstableCourse();
            fact.setReason("believed to be high risk" +
              ($reason.length() > 0 ? " due to " + $reason : ""));
            insert(fact);
end

# RECOMMENDATION: 2. Who performs retinal screening examinations
# Imperative:   2.1 The International Classification of Retinopathy of Prematurity Revised
# should be used to classify, diagram, and record these retinal findings at the time of
examination.
rule "The International Classification of Retinopathy of Prematurity Revisited should be
used to classify, diagram, and record these retinal findings at the time of examination"
    ruleflow-group "rop-imperative"
    when
        # identify patients who are eligible for retinopathy screening
        ROPEligibleFinal(eligible == true)
    then
        ROPImperative fact = new ROPImperative();
        #fact.setPatient($p);
        fact.setDirective("The International Classification of Retinopathy of Prematurity
Revised should be used " +
                          "to classify, diagram, and record these retinal findings at the
time of examination");
        insert(fact);
end

# Imperative:   2.2 Skills and documentation
rule "Skills and documentation"
    ruleflow-group "rop-imperative"
    when
        # identify patients who are eligible for retinopathy screening
        ROPEligibleFinal(eligible == true)
    then
        ROPImperative fact = new ROPImperative();
        #fact.setPatient($p);
        fact.setDirective("Retinal examinations in preterm infants should be performed by
an ophthalmologist " +
```

```
                        "who has sufficient knowledge and experience to enable accurate
identification of the " +
                        "location and sequential retinal changes of ROP.");
        insert(fact);
end

# RECOMMENDATION: 3. Schedule for retinal examination
# Mathematical implementation of table mapping gestational ages to age at first screening
# gestAge -> firstScreen.ageWeekChron
#      22 -> 9
#      23 -> 8
#      24 -> 7
#      25 -> 6
#      26 -> 5
#      27 -> 4
#      28 -> 4
#      29 -> 4
#      30 -> 4
#      31 -> 4
#      32 -> 4
rule "schedule first screen for patients 22 to 26 weeks 6 days gestational age"
    ruleflow-group "rop-schedule"
    when
        # identify patients with gestational age between 22 weeks and 26 weeks 6 days
                $p: Patient($gestAge: gestAge >= 22 && gestAge < 27, $dueDate: dueDate)
                # identify those who are eligible for screening
                ROPEligibleFinal(eligible == true)
    then
        DateTime gest31 = $dueDate.minusWeeks(9);
        ROPSchedule fact = new ROPSchedule();
        fact.setType("first-screening");
        fact.setPriority(2);
        fact.setInterval(new Interval(gest31, gest31.plusDays(7)));
        fact.setReason("Initial screening recommended at age 31 weeks postmenstrual age");
        insert(fact);
end

# provide a safety net rule to cover subjects who are eligible for screening
# but > 33 weeks gestation -- ensures that a first screening is recommended regardless
# of gestational age
rule "schedule first screen for patients 27 to 32 weeks 6 days gestational age"
    ruleflow-group "rop-schedule"
    when
        # identify patients with gestational age 27 weeks or more
                $p: Patient($gestAge: gestAge >= 27, $birthDate: birthDate)
                # identify those who are eligible for screening
                ROPEligibleFinal(eligible == true)
    then
        # first screening at 4 weeks chronologic
        DateTime chron4 = $birthDate.plusWeeks(4);
        ROPSchedule fact = new ROPSchedule();
        fact.setType("first-screening");
        fact.setPriority(2);
        fact.setInterval(new Interval(chron4, chron4.plusDays(7)));
```

```
        fact.setReason("Initial screening recommended at 4 weeks chronologic age");
        insert(fact);
end

# Maintenance challenge:
# Complex, overlapping logic with possibly incomplete coverage exists for scheduling
# ROP examinations.  Propose a set of candidate follow-up intervals, and then select the
longest permitted interval
# specified among the candidates (tolerate risk of "false positives" among the shorter
interval candidates)
# in particular, "disease regression" often justifies a longer follow-up interval

# RECOMMENDATION: 4. Follow-up examinations
rule "followup for findings"
    ruleflow-group "rop-schedule"
    when
        $followup: ROPFollowupCandidate($finding: finding, $interval: interval)
        # for each proposed followup candidate, check to see if a documented finding
        # permits a later followup interval.  Accept the candidate followup if there is no
        # "later" interval found
        not (exists ROPFollowupCandidate(eval (finding == $finding &&
interval.isAfter($interval.getStart()))))
    then
        # Action: Accept this candidate follow-up period and recommend scheduling a
screening
        ROPSchedule fact = new ROPSchedule();
        fact.setType("followup-screening");
        fact.setPriority(3);
        fact.setFinding($finding);
        fact.setInterval($interval);
        fact.setReason($followup.getReason());
        insert(fact);
end

# Verification of coverage of potential findings. Assume that presence of any ROP stage 1
or more requires
# at least SOME follow-up regardless of zone.
#   Zone 1: follow-up specified for stage >=1 in Conditional 4.1, and immature
vascularization in 4.2
#   Zone 2: follow-up specified for stage >=3 in Conditional 4.1, stage == 2 in 4.2, stage
1 in 4.3, and immature vascularization in 4.4
#   Zone 3: follow-up specified for stage >=1 in Conditional 4.4. No coverage for immature
vascularization

# Conditional:  4.1. 1 week or less follow-up
# Logic: If ( (Stage 1 ROP OR Stage 2 ROP) AND Zone I ) OR # implies "OR Stage 3 or more
AND Zone 1"
#             (Stage 3 ROP AND Zone II )
#        Then 1-week or less follow-up
# NOTE: (1) Partial verlap with conditional 4.2 which addresses regressing ROP stage in
zone 1
#       (2) No coverage specified for Stage 3 or more disease in zone 1 (assume included
by this rule -- see "implies" above)
#       (3) Nonsense combinations of stage, zone and maturation may be described by
```

```
clinicians.  As a sanity check, we will
#           make sure there is no assertion that vascularization is complete of stage and
zone are present
rule "1 week or less follow-up"
    ruleflow-group "rop-followup-candidate"
    when
        $finding: ROPFinding($dtFinding: date, zone == 1 && stage >= 1 ||  # include
stages 1 or more
                                               zone == 2 && stage >= 3,    # include
stages 3 or more
                                               completeVascularization == false)
    then
        # Action: 1-week or less follow-up (define as 2-7 days)
        ROPFollowupCandidate fact = new ROPFollowupCandidate();
        fact.setFinding($finding);
        fact.setInterval(new Interval($dtFinding.plusDays(2), $dtFinding.plusDays(7)));
        fact.setReason("Followup in 1 week or less");
        insert(fact);
end

# Conditional:  4.2. 1 to 2 week follow-up
# Logic: If (Immature vascularization AND Zone I AND ROP = FALSE) OR
#           (Stage 2 ROP AND Zone II) OR
#           (Regressing ROP AND Zone I)  # Requires a previous exam -- split into separate
rule
#        Then 1 to 2 week follow-up
# NOTE: stage 0 in zone 1 implies immature vascluarization, but "vascularization complete"
may
# be used erroneously in combination with any zone specification, which is not clinically
valid.
# in this circumstance we will assume that the clinician meant that vascularization is
complete,
# and that the zone specification was erroneous
rule "1 to 2 week follow-up"
    ruleflow-group "rop-followup-candidate"
    when
        $finding: ROPFinding($dtFinding: date, zone == 1 && stage == 0 &&
immatureVascularization == true ||
                                               zone == 1 && stage == 0 &&
completeVascularization == false ||
                                               zone == 2 && stage == 2,
completeVascularization == false)
    then
        # Action: 1 to 2 week follow-up (interpret as 7 to 14 days inclusive)
        ROPFollowupCandidate fact = new ROPFollowupCandidate();
        fact.setFinding($finding);
        fact.setInterval(new Interval($dtFinding.plusDays(7), $dtFinding.plusDays(14)));
        fact.setReason("Followup in 1 to 2 weeks");
        insert(fact);
end

rule "1 to 2 week follow-up regressing zone 1"
    ruleflow-group "rop-followup-candidate"
    when
```

```
        # identify patients with findings in zone 1
        $finding: ROPFinding($dtFinding: date, zone == 1, completeVascularization ==
false)
        # determine if these findings represent regression
        exists (ROPRegressing(finding == $finding))
    then
        # Action: 1 to 2 week follow-up (interpret as 7 to 14 days inclusive)
        ROPFollowupCandidate fact = new ROPFollowupCandidate();
        fact.setFinding($finding);
        fact.setInterval(new Interval($dtFinding.plusDays(7), $dtFinding.plusDays(14)));
        fact.setReason("Followup in 1 to 2 weeks for regressing ROP");
        insert(fact);
end

# Conditional:  4.3. 2 week follow-up
# Logic: If (Stage 1 ROP AND Zone II) OR
#           (Regressing ROP AND Zone II)  # Requires a previous exam -- split into
separate rule
#        Then 2-week follow up
rule "2 week follow-up"
    ruleflow-group "rop-followup-candidate"
    when
        $finding: ROPFinding($dtFinding: date, zone == 2 && stage == 1,
completeVascularization == false)
    then
        # Action: 2 week follow-up (interpret as 10 to 17 days inclusive)
        ROPFollowupCandidate fact = new ROPFollowupCandidate();
        fact.setFinding($finding);
        fact.setInterval(new Interval($dtFinding.plusDays(10), $dtFinding.plusDays(17)));
        fact.setReason("Followup in 2 weeks");
        insert(fact);
end

rule "2 week follow-up regressing zone 2"
    ruleflow-group "rop-followup-candidate"
    when
        # identify patients with findings in zone 2
        $finding: ROPFinding($dtFinding: date, zone == 2, completeVascularization ==
false)
        # determine if these findings represent regression
        exists (ROPRegressing(finding == $finding))
    then
        # Action: 1 to 2 week follow-up (interpret as 7 to 14 days inclusive)
        ROPFollowupCandidate fact = new ROPFollowupCandidate();
        fact.setFinding($finding);
        fact.setInterval(new Interval($dtFinding.plusDays(7), $dtFinding.plusDays(14)));
        fact.setReason("Followup in 2 weeks for regressing ROP");
        insert(fact);
end

# Conditional:  4.4. 2 to 3 week follow-up
# Logic: If (Immature vascularization AND Zone II AND ROP = FALSE) OR
#           ((Stage 1 ROP OR Stage 2 ROP) AND Zone III) OR  # implies "OR Stage 3 or more
and Zone III" which is not covered elsewhere
```

```
#           (Regressing ROP and Zone III) # split
#        Then 2 to 3 week follow-up
# NOTE: No coverage specified for Stage 3 or more disease in zone 3 (assume included by
this rule -- see "implies" above)
# also added safety net criteriea for zone == 0 (implying an unspecified zone), which may
arise since some terminology systems
# encourage document a stage without a zone (e.g. IMO codes)
rule "2 to 3 week follow-up"
    ruleflow-group "rop-followup-candidate"
    when
        $finding: ROPFinding($dtFinding: date, $zone: zone, $stage: stage,
completeVascularization == false,
                             zone == 2 && stage == 0 && immatureVascularization == true ||
                             (zone == 3 || zone == 0) && stage >= 1)
    then
        # Action: 2 to 3 week follow-up (interpret as 14 to 21 days inclusive)
        ROPFollowupCandidate fact = new ROPFollowupCandidate();
        fact.setFinding($finding);
        fact.setInterval(new Interval($dtFinding.plusDays(14), $dtFinding.plusDays(21)));
        fact.setReason("Followup in 2 to 3 weeks");
        insert(fact);
end

rule "2 to 3 week follow-up regressing zone 3"
    ruleflow-group "rop-followup-candidate"
    when
        # identify patients with findings in zone 3
        $finding: ROPFinding($dtFinding: date, $zone: zone, $stage: stage, zone == 3 ||
zone == 0, completeVascularization == false)
        # determine if these findings represent regression
        exists ROPRegressing(finding == $finding)
    then
        # Action: 2 to 3 week follow-up (interpret as 14 to 21 days inclusive)
        ROPFollowupCandidate fact = new ROPFollowupCandidate();
        fact.setFinding($finding);
        fact.setInterval(new Interval($dtFinding.plusDays(14), $dtFinding.plusDays(21)));
        fact.setReason("Followup in 2 to 3 weeks for regressing ROP");
        insert(fact);
end

# TODO: double check this recommendation.  appears that this distills down to
# "any retinopathy with plus disease in zone 1 or 2"

# RECOMMENDATION: 5. New considerations for ablative care
# Conditional:  Ablative treatment initiated
# If (Zone I = TRUE AND ROP = TRUE AND (Stage I OR Stage 2 OR Stage 3) AND Plus Disease
=TRUE) OR
#    (Zone I AND ROP AND NOT (Stage I OR Stage 2 OR Stage 3) AND Plus Disease =FALSE) OR
#    (Zone II = TRUE and (Stage I OR Stage 2) AND Plus Disease =TRUE) Plus Disease Stage I
Stage 2 Stage 3
# Then Ablative treatment
rule "ablative care"
    ruleflow-group "rop-schedule"
    when
```

```
        # look for evidence of retinopathy with plus disease in zone 1 or 2
        $finding: ROPFinding($dtFinding: date, (zone == 1 || zone == 2) && stage >= 1 &&
plusDisease == true, completeVascularization == false)
    then
        # Action: Ablative treatment
                # Description: Treatment should generally be accomplished, when possible,
                # within 72 hours of determination of treatable disease to minimize the
risk of retinal detachment.
        ROPSchedule fact = new ROPSchedule();
        fact.setType("ablation");
        fact.setPriority(1);
        fact.setFinding($finding);
        fact.setInterval(new Interval($dtFinding.plusDays(0), $dtFinding.plusDays(3)));
        fact.setReason("The presence of plus disease in zone " + $finding.getZone() + "
suggests that peripheral" +
                       " ablation, rather than observation, is appropriate");
        insert(fact);
end

# Imperative:   Practitioners involved in the ophthalmologic care of preterm infants
should be
# aware that the retinal findings that require strong consideration of ablative treatment
# were revised recently according to the Early Treatment for Retinopathy of Prematurity
Randomized Trial study.
rule "revised ablation guidelines"
    ruleflow-group "rop-imperative"
    when
        # identify patients who are eligible for retinopathy screening
        ROPEligibleFinal(eligible == true)
    then
        ROPImperative fact = new ROPImperative();
        #fact.setPatient($p);
        fact.setDirective("Practitioners involved in the ophthalmologic care of preterm
infants should be aware " +
                          "that the retinal findings that require strong consideration of
ablative treatment were " +
                          "revised recently according to the Early Treatment for
Retinopathy of Prematurity " +
                          "Randomized Trial study.");
        insert(fact);
end

# RECOMMENDATION: 6. The conclusion of retinal screening exams
# Conditional:  6.1 Exam conclusion finding 1
# Logic: If (Zone III =TRUE) AND NOT (Previous ROP = TRUE AND (Zone I = TRUE OR Zone II =
TRUE))
#        Then Conclusion of acute retinal screening examinations.
# If there is examiner doubt about the zone or if the postmenstrual age is less than 35
weeks, confirmatory examinations may be warranted.
#  Note: implies absence of retinopathy (stage = 0) in zone 3
# also added safety net criteriea for zone == 0 (implying an unspecified zone), which may
arise since some terminology systems
# encourage document a stage without a zone (e.g. IMO codes)
rule "exam conclusion finding 1"
```

```
    ruleflow-group "rop-schedule"
    when
        # Check to see if this conditional has been modified by local policy
        # Do not execute this guideline if a local version of these criteria exist
        not (exists (Localization(guideline == "ROP", conditional == "6.1")))
        # idnetify findings of Zone 3 Stage 0
        $finding: ROPFinding($dtFinding: date, zone == 3 || zone == 0, stage == 0)
        # check to be sure there was not previously zone 1 or 2 disease
        not (exists (ROPFinding ((zone == 1 || zone == 2) && stage >= 1)))
    then
        ROPSchedule fact = new ROPSchedule();
        fact.setType("conclude-screening");
        fact.setPriority(4);
        fact.setFinding($finding);
        fact.setReason("Conclude screening if no retinopathy in zone 3 and no prior " +
                       "retinopathy documented in zone 1 or 2");
        insert (fact);
end

# Conditional:  6.2 Exam conclusion finding 2
# Logic: If Full retinal vascularization = TRUE
#        Then Conclusion of acute retinal screening examinations.
rule "exam conclusion finding 2"
    ruleflow-group "rop-schedule"
    when
        # Check to see if this conditional has been modified by local policy
        # Do not execute this guideline if a local version of these criteria exist
        not (exists (Localization(guideline == "ROP", conditional == "6.2")))
        # Identify patients with complete vascularization
        $finding: ROPFinding(completeVascularization == true)
    then
        ROPSchedule fact = new ROPSchedule();
        fact.setType("conclude-screening");
        fact.setPriority(4);
        fact.setFinding($finding);
        fact.setReason("Conclude screening when vascularization is complete");
        insert (fact);
end

# Conditional:  6.3 Exam conclusion finding 3
# Logic: If Postmenstrual age 45 weeks AND    # equivalent to 5 weeks or 35 days corrected
age
#           NOT ((Stage 3 ROP AND Zone II) OR (ROP AND Zone I) OR ("Worse" ROP)
#        Then Conclusion of acute retinal screening examinations.
# Define "Worse" ROP as increasing stage of ROP in any zone compared to the previous exam
rule "exam conclusion finding 3"
    ruleflow-group "rop-schedule"
    when
        # Check to see if this conditional has been modified by local policy
        # Do not execute this guideline if a local version of these criteria exist
        not (exists (Localization(guideline == "ROP", conditional == "6.3")))
        # identify qualifying findings reported after postmenstrual age 45 weeks
        Patient($dueDate: dueDate)
        $finding: ROPFinding(date >= ($dueDate.plusWeeks(5)),   # postmenstrual 45 weeks
```

```
                            eval(!(zone == 2 && stage >= 3 || zone == 1 && stage >= 1) ),
                            $description: description)
        # check for absence of worsening retinopathy
        not ROPWorsening(finding == $finding)
    then
        ROPSchedule fact = new ROPSchedule();
        fact.setType("conclude-screening");
        fact.setPriority(4);
        fact.setFinding($finding);
        fact.setReason("Conclude screening at postmenstrual age 45 weeks for " +
$description +
                       " with no evidence of worsening retinopathy from prior exams");
        insert (fact);
end

# Conditional:  6.4 Exam conclusion finding 4
# Logic: If Regression of ROP = TRUE
#        Then Conclusion of acute retinal screening examinations.
# NOTE: this needs to be clarified.  Implies that if stage is decreasing then you can
# stop examinations.  For safety, suppress this rule if other rules have proposed a
follow-up
# interval for these findings (implying that it is not the conclusion of screening)
rule "exam conclusion finding 4"
    ruleflow-group "rop-schedule"
    when
        # Check to see if this conditional has been modified by local policy
        # Do not execute this guideline if a local version of these criteria exist
        not (exists (Localization(guideline == "ROP", conditional == "6.4")))
        # identify regressing findings based on decreasing stage of disease between
        # serial exams
        ROPRegressing($finding: finding, $previous: previous)
        # check to see if a follow-up interval was proposed for these findings
        not ROPFollowupCandidate(finding == $finding)
    then
        ROPSchedule fact = new ROPSchedule();
        fact.setType("conclude-screening");
        fact.setPriority(4);
        fact.setFinding($finding);
        fact.setReason("Conclude screening for " + $finding.getDescription() +
                       ($previous != null ? ", which has regressed from " +
$previous.getDescription() : ""));
        insert (fact);
end


# RECOMMENDATION: 7. Communication with the parents
# Imperative:   7.1. Parents should be aware of ROP examinations and should be
#   informed if their child has ROP, with subsequent updates on ROP progression.
rule "Parents should be aware of ROP examinations"
    ruleflow-group "rop-imperative"
    when
        # identify patients who are eligible for retinopathy screening
        ROPEligibleFinal(eligible == true)
    then
```

```
        ROPImperative fact = new ROPImperative();
        #fact.setPatient($p);
        fact.setDirective("Parents should be aware of ROP examinations and should be" +
                          "informed if their child has ROP, with subsequent updates on ROP
progression.");
        insert(fact);
end

# Imperative:   7.2. The possible consequences of serious ROP should be discussed
#   at the time that a significant risk of poor visual outcome develops
# NOTES: this rule could be very helpful for primary care pediatricians if "serious ROP"
# is defined.
rule "consequences of serious ROP should be discussed"
    ruleflow-group "rop-imperative"
    when
        # identify patients who are eligible for retinopathy screening
        ROPEligibleFinal(eligible == true)
    then
        ROPImperative fact = new ROPImperative();
        #fact.setPatient($p);
        fact.setDirective("The possible consequences of serious ROP should be discussed "
+
                           "at the time that a significant risk of poor visual outcome
develops");
        insert(fact);
end

# Imperative:   7.3. Documentation of such conversations with parents in the nurse
#   or physician notes is highly recommended.
rule "documentation of conversations with parents"
    ruleflow-group "rop-imperative"
    when
        # identify patients who are eligible for retinopathy screening
        ROPEligibleFinal(eligible == true)
    then
        ROPImperative fact = new ROPImperative();
        #fact.setPatient($p);
        fact.setDirective("Documentation of conversations with parents in the nurse " +
                          " or physician notes is highly recommended");
        insert(fact);
end

# RECOMMENDATION: 8. Systems recommendations
# Imperative:   If hospital discharge or transfer to another neonatal unit or hospital
#   is contemplated before retinal maturation into zone III has taken place or if the
#   infant has been treated by ablation for ROP and is not yet fully healed, the
#   availability of appropriate follow-up ophthalmologic examination must be ensured,
#   and specific arrangement for that examination must be made before such discharge
#   or transfer occurs.
# NOTE: this could be GEM-cut as a conditional that applies to inpatients only.
# definition for "infant has been treated by ablation for ROP and is not yet fully healed"
# would also be helpful
rule "systems recommendations"
    ruleflow-group "rop-imperative"
```

```
    when
        # identify patients who are eligible for retinopathy screening
        ROPEligibleFinal(eligible == true)
    then
        ROPImperative fact = new ROPImperative();
        #fact.setPatient($p);
        fact.setDirective("If hospital discharge or transfer to another neonatal unit or
hospital " +
          "is contemplated before retinal maturation into zone III has taken place or if
the " +
          "infant has been treated by ablation for ROP and is not yet fully healed, the "
+
          "availability of appropriate follow-up ophthalmologic examination must be
ensured, " +
          "and specific arrangement for that examination must be made before such
discharge " +
          "or transfer occurs.");
        insert(fact);
end

# END OF GUIDELINE

# additional supporting rules to define terms, interpret raw data, or support usability

# Identify last valid (non-missing) observation on or before the evaluation date
rule "find last valid observation"
    ruleflow-group "rop-schedule"
    when
        # find evaluation date (nextDay attribute corresponds to midnight at the end of
the evaluation date)
        Patient($nextDay: nextDay)
        # consider ROP observations that occur before end of day on the evalDate
        $last: ROPFinding($lastDate: date < $nextDay)
        # make sure there are no observations after this date, but on or before evalDate
        not ROPFinding(date < $nextDay, date > $lastDate)
    then
        # Describe last valid observation
        ROPLastValid fact = new ROPLastValid();
        fact.setLast($last);
        # by default, assume last retinopathy screen is NOT in the newborn/NICU period
        fact.setNewborn(false);
        insert(fact);
end

# check to see if the last screening documented corresponds to a screening in the newborn/
NICU period
rule "last valid in newborn period"
    ruleflow-group "rop-schedule"
    when
        # find due date
        Patient($dueDate: dueDate, $birthDate: birthDate)
        # consider ROP observations that occur on or before the due date or within 4 weeks
of the birth as "in the newborn period"
        $fact: ROPLastValid(newborn == false, eval(!last.getDate().isAfter($dueDate) || !
```

```
last.getDate().isAfter($birthDate.plusWeeks(4))))
    then
        # indicate that this is a newborn screening
        modify($fact) { setNewborn(true); }
end

# define regressing ROP as any reduction in stage within the same or greater zone
rule "regressing ROP"
    ruleflow-group "rop-interpret-finding"
    when
        # identify patients with findings in zone 1 to 3
        $finding: ROPFinding(zone >= 1 && zone <= 3)
        # identify prior findings
        # NOTE: this rule will not fire unless a previous ophtho exam exists
        $previous: ROPFinding(date < $finding.date)
        # check to make sure this is the immediate prior exam
        not (exists (ROPFinding (date < $finding.date, date > $previous.date)))
        # check to make sure the retinopathy stage is the same or less in the current exam
        # and that the zone is the same or greater in the current exam (this should always
        # be the case, except when data errors exist)
        eval($finding.getStage() < $previous.getStage() && $finding.getZone() >=
$previous.getZone())
    then
        ROPRegressing fact = new ROPRegressing();
        fact.setFinding($finding);
        fact.setPrevious($previous);
        insert(fact);
end

rule "retract regressing ROP duplicates" extends "regressing ROP"
    ruleflow-group "rop-interpret-finding"
    when
        # if we can prove regressing from comparison to prior observations, then
        # retract any duplicate facts that do not reference a prior observation
        $fact: ROPRegressing(previous == null)
    then
        retract($fact);
end

# assume disease regression if any explicit mention of regressing disease
rule "regressing ROP from description"
    ruleflow-group "rop-describe-finding"
    when
        # identify patients with findings in zone 1 to 3
        $finding: ROPFinding(zone >= 1 && zone <= 3, $date: date)
        # inspect all descriptions for findings on this date
        ROPDescription(date == $date, $text: text)
        # construct a matcher object for regessing disease
        $matcher: Matcher() from Regex.ROP_REGRESSING_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches())
        # verify that no statement of worsening has already been inferred (prevent
duplication of facts)
        not ROPRegressing(finding == $finding)
```

```
    then
        ROPRegressing fact = new ROPRegressing();
        fact.setFinding($finding);
        insert(fact);
end

# define worsening ROP as any increase in stage regardless of zone
rule "worsening ROP"
    ruleflow-group "rop-interpret-finding"
    when
        # identify patients with stage 1 or higher disease
        $finding: ROPFinding(stage >= 1)
        # identify prior findings
        # NOTE: this rule will not fire unless a previous ophtho exam exists
        $previous: ROPFinding(date < $finding.date)
        # check to make sure this is the immediate prior exam
        not (exists (ROPFinding (date < $finding.date, date > $previous.date)))
        # check to make sure the retinopathy stage is higher in the current exam
        eval($finding.getStage() > $previous.getStage())
    then
        ROPWorsening fact = new ROPWorsening();
        fact.setFinding($finding);
        fact.setPrevious($previous);
        insert(fact);
end

rule "retract worsening ROP duplicates" extends "worsening ROP"
    ruleflow-group "rop-interpret-finding"
    when
        # if we can prove worsening from comparison to prior observations, then
        # retract any duplicate facts that do not reference a prior observation
        $fact: ROPWorsening(previous == null)
    then
        retract($fact);
end

# assume disease worsening if any explicit mention of worsening disease
rule "worsening ROP from description"
    ruleflow-group "rop-describe-finding"
    when
        # identify patients with stage 1 or higher disease
        $finding: ROPFinding(stage >= 1, $date: date)
        # inspect all descriptions for findings on this date
        ROPDescription(date == $date, $text: text)
        # construct a matcher object for worsening disease
        $matcher: Matcher() from Regex.ROP_WORSENING_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches())
        # verify that no statement of worsening has already been inferred (prevent
duplication of facts)
        not ROPWorsening(finding == $finding)
    then
        ROPWorsening fact = new ROPWorsening();
        fact.setFinding($finding);
```

```
        insert(fact);
end

# provide a standard description of the ROP stage and zone
rule "narrative description immature retina"
    ruleflow-group "rop-interpret-finding"
    when
        $finding: ROPFinding(immatureVascularization == true, $zone: zone, description ==
null, $resolved: resolved)
    then
        modify($finding) {
            setDescription("Immature retina" + ($zone > 0 ? ", zone " + $zone : "") +
($resolved ? " - resolved" : ""));
        }
end

rule "narrative description complete vascularization"
    ruleflow-group "rop-interpret-finding"
    when
        $finding: ROPFinding(completeVascularization == true, immatureVascularization ==
false, description == null)
    then
        modify($finding) {
            setDescription("Complete vascularization");
        }
end

rule "narrative description stage 1 to 3"
    ruleflow-group "rop-interpret-finding"
    when
        $finding: ROPFinding(completeVascularization == false, immatureVascularization ==
false, $stage: stage > 0, $plus: plusDisease, $zone: zone, description == null, $resolved:
resolved)
    then
        modify($finding) {
            setDescription("Stage " + $stage + ($zone > 0 ? ", zone " + $zone : "") +
($plus ? ", with plus disease" : "") + ($resolved ? " - resolved" : ""));
        }
end

rule "narrative description no stage, but plus present"
    ruleflow-group "rop-interpret-finding"
    when
        $finding: ROPFinding(completeVascularization == false, immatureVascularization ==
false, stage == 0, plusDisease == true, $zone: zone, description == null, $resolved:
resolved)
    then
        modify($finding) {
            setDescription("Plus disease present" + ($zone > 0 ? ", zone " + $zone : "") +
($resolved ? " - resolved" : ""));
        }
end

rule "narrative description unknown stage"
```

```
    ruleflow-group "rop-interpret-finding"
    when
        $finding: ROPFinding(completeVascularization == false, immatureVascularization ==
false, stage == 0, plusDisease == false, $zone: zone, description == null, $resolved:
resolved)
    then
        modify($finding) {
            setDescription("Unknown stage" + ($zone > 0 ? ", zone " + $zone : "") +
($resolved ? " - resolved" : ""));
        }
end

# roll up encounter associated retinopathy descriptors into a single finding
rule "initialize findings"
    ruleflow-group "rop-describe-finding"
    when
        # identify patients who are eligible for retinopathy screening
        $p: Patient()
        ROPEligibleFinal()
        # identify dates of ROP findings
        ROPDescription($date: date)
        # verify that we have not already initialized a finding for this date
            not ROPFinding(date == $date)
    then
        #initialize an empty set of findings for this date
        ROPFinding fact = new ROPFinding();
        fact.setDate($date);
        fact.setStage(0);
        fact.setZone(0);
        fact.setPlusDisease(false);
        fact.setImmatureVascularization(false);
        fact.setCompleteVascularization(false);
        fact.setResolved(false);
        fact.setSource("");
        insert(fact);
end

rule "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        $finding: ROPFinding($date: date, $stage: stage, $zone: zone, $plus: plusDisease,
$immature: immatureVascularization, $complete: completeVascularization, $resolved:
resolved)
        # see if a description of stage 1 disease exists on this date
        ROPDescription(date == $date, $text: text)
    then
        # see dependent rules
end

rule "describe stage 1 disease" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for stage 1 disease
        $matcher: Matcher() from Regex.ROP_STAGE_1_REGEX.matcher($text)
```

```
        # see if it matches
        eval($matcher.matches() && $stage < 1)
    then
        modify($finding) { setStage(1); }
end

rule "describe stage 2 disease" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for stage 2 disease
        $matcher: Matcher() from Regex.ROP_STAGE_2_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && $stage < 2)
    then
        modify($finding) { setStage(2); }
end

rule "describe stage 3 disease" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for stage 3 disease
        $matcher: Matcher() from Regex.ROP_STAGE_3_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && $stage < 3)
    then
        modify($finding) { setStage(3); }
end

rule "describe stage 4 disease" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for stage 4 disease
        $matcher: Matcher() from Regex.ROP_STAGE_4_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && $stage < 4)
    then
        modify($finding) { setStage(4); }
end

rule "describe stage 5 disease" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for stage 5 disease
        $matcher: Matcher() from Regex.ROP_STAGE_5_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && $stage < 5)
    then
        modify($finding) { setStage(5); }
end

rule "describe zone 1 disease" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for zone 1 disease
```

```
        $matcher: Matcher() from Regex.ROP_ZONE_1_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && $zone < 1)
    then
        modify($finding) { setZone(1); }
end

rule "describe zone 2 disease" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for zone 2 disease
        $matcher: Matcher() from Regex.ROP_ZONE_2_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && $zone < 2)
    then
        modify($finding) { setZone(2); }
end

rule "describe zone 3 disease" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for zone 3 disease
        $matcher: Matcher() from Regex.ROP_ZONE_3_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && $zone < 3)
    then
        modify($finding) { setZone(3); }
end

rule "describe plus disease" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for plus disease
        $matcher: Matcher() from Regex.ROP_PLUS_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && !$plus)
    then
        modify($finding) { setPlusDisease(true); }
end

rule "describe immature retina" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for immature retina
        $matcher: Matcher() from Regex.ROP_IMMATURE_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && !$immature)
    then
        modify($finding) { setImmatureVascularization(true); }
end

rule "describe complete vascularization" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
```

```
        # construct a matcher object for complete vascularization
        $matcher: Matcher() from Regex.ROP_COMPLETE_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && !$complete)
    then
        modify($finding) { setCompleteVascularization(true); }
end

rule "describe resolved finding" extends "describe individual findings"
    ruleflow-group "rop-describe-finding"
    when
        # construct a matcher object for complete vascularization
        $matcher: Matcher() from Regex.ROP_RESOLVED_REGEX.matcher($text)
        # see if it matches
        eval($matcher.matches() && !$resolved)
    then
        modify($finding) { setResolved(true); }
end

# Describe best source of documentation, prioritize in the following order
# "Diagnosis", "Problem", "MedicalHistory", "BirthHistory", or "Narrative"
rule "describe diagnosis source of finding"
    ruleflow-group "rop-describe-finding"
    when
        $finding: ROPFinding($date: date, source != "Diagnosis")
        # see if a description retinopathy is found in an encounter diagnosis
        ROPDescription(date == $date, $e: encounter, $source: source == "Diagnosis")
    then
        modify($finding) { setSource($source), setUrl($e.getUrl()) }
end

rule "describe problem list source of finding"
    ruleflow-group "rop-describe-finding"
    when
        $finding: ROPFinding($date: date, source != "Diagnosis", source != "Problem")
        # see if a description retinopathy is found on the problem list
        ROPDescription(date == $date, $source: source == "Problem")
    then
        modify($finding) { setSource($source); }
end

rule "describe medical history source of finding"
    ruleflow-group "rop-describe-finding"
    when
        $finding: ROPFinding($date: date, source != "Diagnosis", source != "Problem",
source != "MedicalHistory")
        # see if a description retinopathy is found in the medical history
        ROPDescription(date == $date, $source: source == "MedicalHistory")
    then
        modify($finding) { setSource($source); }
end

rule "describe birth history source of finding"
    ruleflow-group "rop-describe-finding"
```

```
    when
        $finding: ROPFinding($date: date, source != "Diagnosis", source != "Problem",
source != "MedicalHistory", source != "BirthHistory")
        # see if a description retinopathy is found in the birth history
        ROPDescription(date == $date, $source: source == "BirthHistory")
    then
        modify($finding) { setSource($source); }
end

# use narrative as the source as a last resort
rule "describe narrative source of finding"
    ruleflow-group "rop-describe-finding"
    when
        $finding: ROPFinding($date: date, source == "")
        # see if a description retinopathy is found in an encounter diagnosis
        ROPDescription(date == $date, $e: encounter, $source: source == "Diagnosis")
    then
        modify($finding) { setSource($source), setUrl($e.getUrl()) }
end

# identify diagnosis descriptors that may identify stage / zone of disease
rule "extract ROP diagnoses"
    ruleflow-group "rop-extract-finding"
    when
        # identify patients who are eligible for retinopathy screening
        $p: Patient()
        ROPEligibleFinal()
        # identify encounter associated diagnoses that may describe retinopathy
        $e: Encounter() from $p.getEncounters()
        # examine diagnoses with ICD9 code matching 362.2
            $d: Diagnosis($source: source, icd9 matches "362\\.2.*") from
$e.getDiagnoses()
    then
        ROPDescription fact = new ROPDescription();
        fact.setSource($source);
        fact.setEncounter($e);
        fact.setDate($e.getInstant().toDateMidnight());
        fact.setText($d.getDescription());
        insert(fact);
end

# identify problem list and medical history text that may identify stage / zone of disease
rule "extract ROP problem list and medical history"
    ruleflow-group "rop-extract-finding"
    when
        # identify patients who are eligible for retinopathy screening
        $p: Patient($birthDate: birthDate, $dueDate: dueDate)
        ROPEligibleFinal()
        # identify patient associated problem list diagnoses that may describe retinopathy
            Diagnosis($text: description, $source: source, icd9 matches "362\\.2.*",
$status: status) from $p.getDiagnoses()
    then
        # see dependent rules
end
```

```
# identify problem list and medical history text that may identify stage / zone of disease
rule "extract ROP problem list" extends "extract ROP problem list and medical history"
    ruleflow-group "rop-extract-finding"
    when
        eval($source.equals("Problem") && !$status.matches("(?ism).*(?:resolve|del).*"))
    then
        ROPDescription fact = new ROPDescription();
        fact.setSource($source);
        fact.setText($text);
        insert(fact);
end

# identify problem list and medical history text that may identify stage / zone of disease
# resolved diagnoses are considered implicitly to be part of medical history
rule "extract ROP medical history" extends "extract ROP problem list and medical history"
    ruleflow-group "rop-extract-finding"
    when
        eval($source.equals("MedicalHistory") || $status.matches("(?ism).*(?:resolve|
del).*"))
    then
        # entries on medical history are implicitly resolved
        ROPDescription fact = new ROPDescription();
        fact.setSource("MedicalHistory");
        fact.setText("resolved " + $text);
        insert(fact);
end

# check to see if 31 weeks gestation is later than proposed date for this
# parse retinopathy narratives to identify stage / zone descriptors from narrative
documentation
# associated with encounters
rule "extract ROP narrative"
    ruleflow-group "rop-extract-finding"
    when
        # identify patients who are eligible for retinopathy screening
        $p: Patient()
        ROPEligibleFinal()
        # identify narrative fragments that appear to describe retinopathy
        $e: Encounter() from $p.getEncounters()
        # check for encounters that
        $narr: Narrative($text: text) from $e.getNarratives()
        # construct a matcher object to classify number of patients seen
        $matcher: Matcher() from Regex.ROP_REGEX.matcher($text)
        # determine if the pattern matches
        eval($matcher.matches())
    then
        ROPDescription fact = new ROPDescription();
        fact.setSource("Narrative");
        fact.setEncounter($e);
        fact.setDate($e.getInstant().toDateMidnight());
        fact.setText($matcher.group(1));
        insert(fact);
end
```

```
# parse retinopathy narratives to identify stage / zone descriptors from birth history
documentation
rule "extract ROP birth history"
    ruleflow-group "rop-extract-finding"
    when
        # identify patients who are eligible for retinopathy screening
        $p: Patient($birthDate: birthDate, $birthHistory: birthHistory)
        ROPEligibleFinal()
        # construct a matcher object to classify newborn retinopathy results
        $matcher: Matcher() from Regex.ROP_REGEX.matcher($birthHistory.getBirthComment())
        # determine if the pattern matches
        eval($matcher.matches())
    then
        ROPDescription fact = new ROPDescription();
        fact.setSource("BirthHistory");
        fact.setText($matcher.group(1));
        insert(fact);
end

# attempt to find an explicit date in the description
rule "identify date from text"
    ruleflow-group "rop-extract-finding"
    when
        $fact: ROPDescription($date: date != null, $text: text)
        # see if we can find a date in the string
        $newDate: DateMidnight() from FuzzyDate.dateFromString($text)
        # see if this new date occurs after the currently documented date
        eval($newDate.isAfter($date))
    then
        # assign the new later date to the fact
        modify($fact) { setDate($newDate); }
end

# screening is performed at a minimum of 31 weeks post menstrual
rule "assume 31 weeks post menstrual age"
    ruleflow-group "rop-extract-finding"
    when
        Patient($dueDate: dueDate)
        # calculate 4 weeks chronologic age
        $newDate: DateMidnight() from $dueDate.minusWeeks(9).toDateMidnight()
        # see if this new date occurs after the currently documented date
        $fact: ROPDescription(eval(date == null || $newDate.isAfter(date)))
    then
        # assign the new later date to the fact
        modify($fact) { setDate($newDate); }
end

# screening is performed at a minimum of 4 weeks chronologic age
rule "assume 4 weeks chronologic age"
    ruleflow-group "rop-extract-finding"
    when
        Patient($birthDate: birthDate)
        # calculate 4 weeks chronologic age
```

```
        $newDate: DateMidnight() from $birthDate.plusWeeks(4).toDateMidnight()
        # see if this new date occurs after the currently documented date
        $fact: ROPDescription(eval(date == null || $newDate.isAfter(date)))
    then
        # assign the new later date to the fact
        modify($fact) { setDate($newDate); }
end

# suppress past medical history events if there are any other findings described on the
same date
# from a higher priority source (e.g.
rule "suppress past medical history"
    ruleflow-group "rop-clean-finding"
    when
        # identify descriptions of ROP that arise from medical history
        $fact: ROPDescription($date: date, source == "MedicalHistory")
        # check to see if a description of ROP from a preferred source exists on the same
date
        exists ROPDescription(date == $date, source == "Problem" || source == "Diagnosis")
    then
        # remove the medical history fact from working memory
        retract($fact);
end

# final recommendations -- execute after all other rules have fired
rule "ROP final recommendations"
    ruleflow-group "rop-recommendation"
    when
        # include final eligibility information
        $e: ROPEligibleFinal()
        # include any imperatives that affect eligibility or schedule
        $imperatives: ArrayList() from collect (ROPImperative())
        # include all ROP findings that have been documented
        $findings: ArrayList() from collect (ROPFinding())
        # include all ROP screening and treatment schedule information # TODO
        $schedule: ArrayList() from collect (ROPSchedule())
    then
        ROPRecommendation fact = new ROPRecommendation();
        #fact.setPatient($p);
        fact.setEligible($e);
        fact.setImperatives($imperatives);
        fact.setFindings($findings);
        fact.setSchedule($schedule);
        fact.setSummaryLast("");
        fact.setSummaryFollowup("");
        fact.setSummaryAction("");
        fact.setSummaryStatement("");
        fact.setSummaryBriefStatement("");
        fact.setFlag(false);
        fact.setResourceUrl("");
        insert(fact);
end

# summarize no rop documentation found
```

```
rule "no documentation found"
    ruleflow-group "rop-recommendation"
    when
        # identify recommendations where no summary statement has been made
        $fact: ROPRecommendation(summaryLast == "")
        # verify eligibility for screening
        ROPEligibleFinal(eligible == true)
        # verify no rop findings documented
        not ROPLastValid()
    then
        modify($fact) {
            setSummaryLast("<b>No Documentation Found</b>")
        }
end

# describe the last rop result
# there are four possibilities: screening may from the newborn perior or from later, and
# follow-up schedule may exist or not exist
rule "describe last rop"
    ruleflow-group "rop-recommendation"
    when
        # identify recommendations where no summary statement has been made regarding
prior screening
        $fact: ROPRecommendation(summaryLast == "")
    then
        # see dependent rules
end

rule "describe newborn rop" extends "describe last rop"
    ruleflow-group "rop-recommendation"
    when
        # see if last valid observation is the newborn rop screen
        ROPLastValid($last: last, newborn == true)
    then
        # avoid being overly precise about when newborn screening was done, since the date
is rarely documented
        modify($fact) {
            setSummaryLast($last.getDescription()),
            setResultUrl($last.getUrl())
        }
end

rule "describe subsequent rop screen" extends "describe last rop"
    ruleflow-group "rop-recommendation"
    when
        # see if last valid observation is a follow-up rop screen
        ROPLastValid($last: last, newborn == false)
    then
        # describe the date and result of this rop screen
        modify($fact) {
            setSummaryLast($last.getDescription() + " (" +
                           FuzzyDate.monthDayYearBrief($last.getDate()) + ")"),
            setResultUrl($last.getUrl())
        }
```

```
end

rule "describe follow-up"
    ruleflow-group "rop-recommendation"
    when
        # identify recommendations where no summary statement has been made regarding
prior screening
        $fact: ROPRecommendation(summaryFollowup == "")
        # find last valid observation
        ROPLastValid($last: last)
        # see what follow-up is recommended
        ROPSchedule(finding == $last, $priority: priority, $reason: reason)
        # verify that no higher priority schedule exists for this finding
        not ROPSchedule(finding == $last, priority < $priority)
    then
        # specify reason for follow-up
        modify($fact) {
            setSummaryFollowup($reason)
        }
end

# Suggest strategies for how to document resolution of retinopathy screening
# depending on where documentation was found

rule "retinopathy next steps"
    ruleflow-group "rop-recommendation"
    when
        # identify recommendations where no summary statement has been made
        $fact: ROPRecommendation(summaryAction == "")
        # identify last valid screening
        ROPLastValid($last: last)
        # check to see if follow-up was recommended
        $schedule: ROPSchedule($interval: interval, finding == $last, type != "conclude-
screening")
    then
        # see dependent rules
end

#If Patient has history of ROP documented on PROBLEM LIST
#AND
#ROP is clinically resolved now
#THEN
#RESOLVE on Problem List, and move to past medical history (acceptable in birth history)
#If Patient has history of ROP documented on PROBLEM LIST
#AND
#ROP is unresolved
#THEN
#Updated Problem List comments (ideally add to progress note)
rule "retinopathy on problem list" extends "retinopathy next steps"
    ruleflow-group "rop-recommendation"
    when
        # check to see if source is problem list
        eval($last.getSource().equals("Problem"))
    then
```

```
        modify($fact) {
            setSummaryAction("Update comment on problem list or move to past medical
history with date resolved as a comment; document plan in progress note"),
            setFlag(true),
            setActionProblemList("Update problem list")
        }
end

#If Patient has history of ROP documented on BIRTH HISTORY, encounter DIAGNOSIS or only in
NARRATIVE notes
#AND
#ROP is clinically resolved now
#THEN
#add to past medical history (acceptable to edit birth history)
#If Patient has history of ROP documented on BIRTH HISTORY, encounter DIAGNOSIS or only in
NARRATIVE notes
#AND
#ROP is unresolved
#THEN
#Add to Problem List with clarifying comments (ideally add to progress note)
rule "retinopathy on birth history" extends "retinopathy next steps"
    ruleflow-group "rop-recommendation"
    when
        # check to see if source is birth history
        eval($last.getSource().equals("BirthHistory") ||
$last.getSource().equals("Narrative")  || $last.getSource().equals("Diagnosis"))
    then
        modify($fact) {
            setSummaryAction("Document active ROP on problem list or as past medical
history with date resolved as a comment; document plan in progress note"),
            setFlag(true),
            setActionProblemList("Update problem list"),
            setActionPastMedicalHx("Update medical history")
        }
end

#If Patient has history of ROP documented on PAST MEDICAL HISTORY ONLY (implies that
disease has resolved)
#AND
#ROP is clinically resolved now
#THEN
#suggest clarifying comment of ?resolved? on med history entry
rule "retinopathy on past medical history" extends "retinopathy next steps"
    ruleflow-group "rop-recommendation"
    when
        # check to see if source is medical history
        eval($last.getSource().equals("MedicalHistory"))
    then
        modify($fact) {
            setSummaryAction("Clarify ROP as resolved on past medical history"),
            setFlag(true),
            setActionPastMedicalHx("Update medical history")
        }
end
```

```
# see if prior result indicates that screening is complete
rule "retinopathy screening complete"
    ruleflow-group "rop-recommendation"
    when
        # identify recommendations where no summary statement has been made
        $fact: ROPRecommendation(summaryAction == "")
        # identify last valid screening
        ROPLastValid($last: last)
        # verify no follow-up was recommended
        not ROPSchedule(finding == $last, type != "conclude-screening")
    then
        modify($fact) { setSummaryAction("No further screening routinely recommended") }
end

# if no prior result and initial screening was recommended
rule "next steps no initial screen documented"
    ruleflow-group "rop-recommendation"
    when
        $p: Patient($evalDate: evalDate)
        # identify recommendations where no summary statement has been made
        $fact: ROPRecommendation(summaryAction == "")
        # describe if prior no rop findings documented
        not ROPLastValid()
    then
        # see dependent rules
end

rule "next steps initial screening expected" extends "next steps no initial screen
documented"
    ruleflow-group "rop-recommendation"
    when
        # verify initial screening (not associated with any finding) recommended
        ROPSchedule(finding == null, type != "conclude-screening")
    then
        modify($fact) {
            setSummaryAction("Please document active ROP on problem list, resolved ROP as
past medical history, or absence of ROP on birth history"),
            setFlag(true),
            setActionProblemList("Update problem list"),
            setActionBirthHx("Update birth history"),
            setActionPastMedicalHx("Update medical history")
        }
end

rule "no initial screening expected" extends "next steps no initial screen documented"
    ruleflow-group "rop-recommendation"
    when
        # verify no initial screening recommended
        not ROPSchedule(finding == null, type != "conclude-screening")
    then
        modify($fact) { setSummaryAction("No screening routinely recommended") }
end
```

```
# produce overall summary statement
# there may be some gaps in coverage of last/follow-up/action summary statements
rule "overall summary"
    ruleflow-group "rop-summary-statement"
    when
        $fact: ROPRecommendation($last: summaryLast, $followup: summaryFollowup, $action:
summaryAction,
                                 summaryLast != "" || summaryFollowup != "",
summaryStatement == "")
        # verify that summary should not be hidden
        not Localization(guideline == "ROP", hideGuideline == true)
    then
        # provide overall summary statement for display in user-interface
        String summary = $last;
        if($followup.length() > 0) {
            if(summary.length() > 0) { summary += "<br />"; }
            summary += "AAP: <i>" + $followup + "</i> (or as directed by ophthalmology)";
        }
        String summaryBrief = summary;
        if($action.length() > 0) {
            if(summary.length() > 0) { summary += "<br />"; }
            summary += $action;
        }
        # for now only show the abbreviated summary
        modify($fact) { setSummaryStatement(summaryBrief),
setSummaryBriefStatement(summaryBrief) }
end

# produce overall summary statement
# there may be some gaps in coverage of last/follow-up/action summary statements
rule "visible for pilot only"
    ruleflow-group "localization"
    when
        $p: Patient()
                # verify that this is not a department that may participate as a pilot
site (at least one pilot user)
                not (Identifier(
                    id == "98109512" || # CHESTNUT HILL
                    id == "89296012" || # FACULTY PRACTICE
                    id == "10605012" || # MARKET STREET
                    id == "50601012" || # SOUTH PHILADELPHIA
                    id == "89369021" || # NEONATAL FOLLOW-UP WOOD
                    id == "80368021",   # NEONATAL FOLLOW-UP EXTON
                    type == Identifier.DEPARTMENT_ID) from $p.getIdentifiers())
    then
        Localization fact = new Localization("ROP");
        fact.setHideGuideline(true);
        insert(fact);
end

query "Recommendations"
    recommendation: ROPRecommendation();
end
```