Appendix B

Success Rule and GAC Propagator In Details

B.1 Frequency Success Rule In Details

In this section I will explain my frequency success rule and the propagators that use it. These propagators are simply obtained by adding the frequency success rule to the propagators precedently presented. These propagators try to check if they can return success only when some specific conditions that I will present make this success check less time consuming and relevant.

Rule 7 (***frequency success rule***)**.** $Freq(I(D) \cup unbound) \geq \theta \Rightarrow return\ Success$

correctness of returning Success. As when I return Success, the itemset $I(D) \cup unbound$ is frequent and for any fixed D' stronger than D, $I(D') \subseteq (I(D) \cup unbound)$, for all these D' I(D') is frequent too. □

The implementation uses the *columnsSuccess* data structure which is defined as follows:

columnsSuccess is a table which is such that $\forall\ i\ \in\ \mathcal{I}$ columnsSuccess(i) $= \bigcap_{j\ \in\ [i+1,nItems]} \mathcal{T}(j)$.

The implementation also uses the two rint $nBound$ and $maxBound$ defined as follows: $\boldsymbol{nBound} = |bound|$ and $\boldsymbol{maxBound} = max_{i\ \in\ bound}\ i$.

Let's define a **success check** as the computation of the predicate $|columnsSuccess(nBound) \cap \mathcal{T}(I(D))| \geq \theta$.
I make a success check only if $\boldsymbol{nBound = maxBound}$ because otherwise the condition $|columnsSuccess(nBound) \cap \mathcal{T}(I(D))| \geq \theta$ is different than $Freq(I(D) \cup unbound) \geq \theta$. If I would like to check whether $Freq(I(D) \cup unbound) \geq \theta$ when $nBound \neq maxBound$, I should check whether $|(\bigcap_{i\ \in\ unbound\ s.t.\ i<maxBound} \mathcal{T}(i)) \cap columnsSuccess(maxBound) \cap \mathcal{T}(I(D))| \geq \theta$ Which would take too much time to compute. This is why I enforce the frequency success rule only when $nBound = maxBound$.

Obviously It is useless to make a success check whenever |columnsSuccess($nBound$)|<θ. This is why I make a success check only if $\boldsymbol{nBound \geq trySuccess}$ with trySuccess defined as follows:

trySuccess $= min_{i\ \in\ \mathcal{I}\ s.t.\ |columnsSuccess(i)| \geq \theta}\ i$.

As it is redundant to make a success check if every variable that have been bound since the last success check have been bound to 1, I allow to perform a success check only if **there has been a bind to 0 since the last success check**. This condition is verified thanks to the rboolean thereHasBeenABindTo0SinceLastSuccessCheck.
Finally, I make a success check only if $\boldsymbol{nBound < nItems - 1}$ because if there is only one variable to bind left and if we can return success then we make the same work as in the bind to 0 branch (no work) + the bind to 1 branch where we compute the same intersection.
If we can not return success, we do twice the same intersection once for the success check and once in the bind to 1 branch.
So there must be at least two variables unbound left for this test to enhance the time efficiency of my algorithm.

I created the three propagators BasicFrequencyWithSuccess, ReifiedFrequencyWithSuccess and ReifiedFrequencyWithSuccessAndClosure by adding the frequency success rule to the three propagators already presented. Each of these propagators perform a success check under the conditions explained above and return success when the predicate is true. Algorithm 12 is the pseudo code of the ReifiedFrequencyWithSuccess propagator.

Algorithm 12 ReifiedFrequencyWithSuccess

```
1: coverChanged = false
2: for i ← unbound do
3:     if I_i = 0 then $ if I_i is bound to 0
4:         unbound.remove(i), nBound +=1
5:         if i > maxBound then
6:             maxBound = i
7:     else if I_i = 1 then
8:         coverChanged = coverage.intersectionWith(V_i), unbound.remove(i)
9:         nBound +=1
10:        if i > maxBound then
11:            maxBound = i
12: if coverChanged then
13:     frequency = coverage.cardinality()
14:     if frequency < θ then $ failure of frequency rule
15:         return Failure
16:     for j ← unbound do $ reified frequency filtering rule
17:         freqWithIt = coverage.cardinalityIntersectionWith(V_j)
18:         if freqWithIt < θ then
19:             unbound.remove(i), I_j = 0, nBound +=1
20:             if i > maxBound then
21:                 maxBound = i
22:             thereHasBeenABindTo0SinceLastSuccessCheck.setTrue()
23: else
24:     thereHasBeenABindTo0SinceLastSuccessCheck.setTrue()
25:
    $ conditions to allow a success check
26: if nBound = maxBound && nBound ≥ trySuccess && nBound < nItems −
    1 && thereHasBeenABindTo0SinceLastSuccessCheck.value then $perform a success
    check
27:     if coverage.cardinalityIntersectionWith(columnsSuccess(nBound)) ≥ θ then
28:         return Success
29:     thereHasBeenABindTo0SinceLastSuccessCheck.setFalse()
```

B.2 GAC propagator

In this section I will present a propagator which is General Arc Consistent (GAC). This propagator is similar to the one presented in but avoids some redundant operations.

GAC filtering rule

Rule 8 (***simple GAC filtering rule***)**.** *Bind to 0 any unbound variable* I_u *s.t.* $\exists\ b0\ \in\ boundTo0$ *s.t.* $b0\ \in\ Closure(I(D) \cup u)$.

$$\forall\ u\ \in\ unbound,\ I_{b0}\ \in\ boundTo0:$$

$$Freq(I(D) \cup u \cup b0) = Freq(I(D) \cup u) \Rightarrow I_u = 0$$

To see the correctness of this rule we just have to see that the failure of closedness property applies to every D' s.t. D'$(I_u) = 1$.
The condition $Freq(I(D) \cup u \cup b0) = Freq(I(D) \cup u)$ can be expressed in multiple different ways as: $Freq(I(D) \cup u \cup b0) = Freq(I(D) \cup u) \Leftrightarrow \mathcal{T}(I(D)) \cap \mathcal{T}(u) \cap \mathcal{T}(b0) = \mathcal{T}(I(D)) \cap \mathcal{T}(u) \Leftrightarrow (\mathcal{T}(I(D)) \cap \mathcal{T}(u)) \setminus \mathcal{T}(b0) = \emptyset$, this is the condition GAC rule 1 used in line 30 of the Algorithm13, $\Leftrightarrow (\mathcal{T}(I(D)) \setminus \mathcal{T}(b0)) \cap \mathcal{T}(u) = \emptyset$, this is the condition GAC rule 2 used in line 42 of the Algorithm13.
Thanks to the non closure inclusion of infrequent items property I can rewrite the GAC filtering rule as follows:

$$\forall\ u\ \in\ unbound,\ I_{b0}\ \in\ boundTo0Frequent:$$

$$Freq(I(D) \cup u \cup b0) = Freq(I(D) \cup u) \Rightarrow I_u = 0$$

I can remove redundant operations for the failure of closedness and the GAC filtering rule by not checking those conditions for the items bound to 0 by the GAC filtering rule. Let **boundByGAC** be the set of variables bound by the GAC filtering rule and **testGAC** $= unbound \cup boundTo0Frequent \setminus boundByGAC$, we have:

$$\forall\ rem\ \in\ boundByGAC,\ \exists\ b0\ \in\ boundToOFrequent \setminus boundByGAC$$

$$s.t.\ \mathcal{T}(I(D)) \cap \mathcal{T}(rem) \cap \mathcal{T}(b0) = \mathcal{T}(I(D)) \cap \mathcal{T}(rem)$$

Reformulation of the failure of closedness rule: As at any further call of the propagator, if $rem\ \in\ boundByGAC$ is s.t. $Freq(I(D) \cup rem) = Freq(I(D))$ then as $\mathcal{T}(I(D)) \cap \mathcal{T}(b0) \supseteq \mathcal{T}(I(D)) \cap \mathcal{T}(rem)$ we have $Freq(I(D) \cup b0) = Freq(I(D))$, so it is redundant to keep rem for the failure of closedness rule which can therefore be rewritten in the following equivalent formulation:

Rule 9 (***GAC failure of closedness***)**.**

$$\forall\ I_i\ \in\ boundToOFrequent \setminus boundByGAC:$$

$$Freq(I(D) \cup i) = Freq(I(D)) \Rightarrow return\ Failure$$

reformulation of the GAC filtering rule: As at any further call of the propagator, if $\exists\, u\ \in\ unbound\ s.t.\ \mathcal{T}(I(D)) \cap \mathcal{T}(rem) \cap \mathcal{T}(u) = \mathcal{T}(I(D)) \cap \mathcal{T}(u)$ then as $\mathcal{T}(I(D)) \cap \mathcal{T}(u) \cap \mathcal{T}(b0) = \mathcal{T}(I(D)) \cap \mathcal{T}(rem) \cap \mathcal{T}(u) \cap \mathcal{T}(b0) = \mathcal{T}(I(D)) \cap \mathcal{T}(rem) \cap \mathcal{T}(u) = \mathcal{T}(I(D)) \cap \mathcal{T}(u)$. It is redundant to keep rem for the GAC filtering rule which can therefore be rewritten in the following equivalent formulation:

Rule 10 (***GAC filtering rule***)**.**

$$\forall\ u\ \in\ unbound,\ I_{b0}\ \in\ boundToOFrequent \setminus boundByGAC :$$

$$Freq(I(D) \cup u \cup b0) = Freq(I(D) \cup u) \Rightarrow I_u = 0$$

propagator 5 (***GAC propagator Algorithm 13***)**.** *Enforces the following rules:*

1. *failure of frequency rule.*
2. *GAC failure of closedness rule.*
3. *reified frequency filtering rule.*
4. *closure filtering rule.*
5. *GAC filtering rule.*

As every reversible structures are attached to the reversible context **s**, in Algorithm13, when I do s.pushState(), some modification on my reversible structures, s.pop(), it is as if every modification done between s.pushState() and s.pop() didn't appeared. The states of my reversible structures after s.pop() are the same as just before s.push()

General Arc Consistency Of The GAC Propagator

In they demonstrate that when we enforce the frequency and simple closedness failure rules and then the reified frequency, closure and simple GAC filtering rule we obtain a GAC propagator. As I showed that the simple closedness failure rule is equivalent to the GAC failure failure of closedness rule and the simple GAC filtering rule is equivalent to the GAC filtering rule, my propagator is also GAC. So my propagagor is equivalent to their propagator but it avoids redundant work.

Algorithm 13 GACpropagator

```
1: coverChanged = false, newVariablesBoundTo0 = ∅
2: for i ← testGAC do $ testGAC = unbound ∪ boundTo0Frequent \ boundByGAC
3:     if I_i = 1 then
4:         coverChanged=coverage.intersectionWith(V_i)
5:         testGAC.remove(i)
6:     else if I_i = 0 then
7:         if !boundTo0(i) then
8:             newVariablesBoundTo0 = newVariablesBoundTo0 ∪ i
9:             boundTo0(i) = true
10: if coverChanged then
11:     frequency = coverage.cardinality()
12:     if frequency < θ then $ failure of frequency rule
13:         return Failure
14:     for i ← testGAC do $GAC failure of closedness rule
15:         if I(i) = 0 then
16:             freqWithIt = coverage.cardinalityIntersectionWith(V_i)
17:             if freqWithIt == frequency then
18:                 return Failure
19:             else if freqWithIt < θ then
20:                 testGAC.remove(i)
21:     for i ← testGAC do
22:         if I_i = U then
23:             freqWithIt = coverage.cardinalityIntersectionWith(V_i)
24:             if freqWithIt < θ then $ reified frequency filtering rule
25:                 testGAC.remove(i), I_i = 0
26:                 boundTo0(i) = true
27:             if freqWithIt == frequency then $ closure filtering rule
28:                 testGAC.remove(i), I_i = 1
29:     for itUnBound ← testGAC do $ GAC rule 1
30:         if I_itUnBound = U then
31:             s.pushState(), bindItUnBound = false
32:             coverage.intersectionWith(V_itUnBound)
33:             for itBoundTo0Frequent ← testGAC do
34:                 if I_itBoundTo0Frequent = 0 && !bindItUnBound then
35:                     if coverage.isIncludedIn(V_itBoundTo0Frequent) then
36:                         bindItUnBound = true
37:             s.pop()
38:             if bindItUund then
39:                 I_itUnBound = 0, boundTo0(itUnBound) = true
40:                 testGAC.remove(itUnBound)
41: else $ GAC rule 2
42:     for itBoundTo0Frequent ← newVariablesBoundTo0 do
43:         s.pushState(), variablesToBound = ∅
44:         coverage.remove(V_itBoundTo0Frequent)
45:         for itUnBound ← testGAC do
46:             if I_itUnBound = U then
47:                 if !coverage.intersect(V_itUnBound) then
48:                     variablesToBound = variablesToBound ∪ itUnBound
49:         s.pop()
50:         for i ← variablesToBound do
51:             I_i = 0, boundTo0(i) = true
52:             testGAC.remove(i)
```