```
adjm-edge-accum-properties  THM

⊢  ∀M:AdjMatrix, T:Type, s:T, x:Vertices(adjm-graph(M))
|   , f:(T→Vertices(adjm-graph(M))→T).
|   ∃L:Vertices(adjm-graph(M)) List.
|   (∀y:Vertices(adjm-graph(M)). x-adjm-graph(M)->y ⟺ (y ∈ L))
|   & adjm-edge-accum(M;s',x'.f(s',x');s;x)
|     =
|     list_accum(s',x'.f(s',x');s;L)
| by (Unfolds [‘adjm-graph‘;‘adjm-edge-accum‘] 0) THEN (Reduce 0)
|     THEN
|     (InstConcl [filter(M.adj(x);upto(0;M.size))])
|     THEN
|     (Analyze 0)
|
|\
| 1. M : AdjMatrix
| 2. T : Type
| 3. T
| 4. x : ℕM.size
| 5. T→ℕM.size→T
| ⊢  ∀y:ℕM.size.
| |  x-<vertices = ℕM.size,
| |    edges = {p:(ℕM.size×ℕM.size)| M.adj(1of(p),2of(p)) },
| |    incidence = λe.e>->y
| |  ⟺
| |  (y ∈ filter(M.adj(x);upto(0;M.size)))
| | by (Analyze 0) THEN (Unfold ‘edge‘ 0) THEN (Reduce 0)
| |    THEN
| |    (RWO
| |     Thm* ∀P:(T→𝔹), L:T List, x:T.
| |           (x ∈ filter(P;L)) ⟺ (x ∈ L) & P(x)
| |     {member_filter}
| |     0)
| |    THEN
| |    ExRepD
| |
| |\
| | 1. M : AdjMatrix
| | 2. T : Type
| | 3. T
| | 4. x : ℕM.size
| | 5. T→ℕM.size→T
| | 6. y : ℕM.size
| | 7. e : {p:(ℕM.size×ℕM.size)| M.adj(1of(p),2of(p)) }
| | 8. e = <x,y>
| | ⊢  (y ∈ upto(0;M.size))
| | | by AssertBY (y ∈ upto(0;M.size))
| | |    (RWO
| | |     Thm* ∀i,j,k:ℕ. (k ∈ upto(i;j)) ⟺ i≤k & k<j    {member_upto}
| | |     0)
| | |
| | 1. M : AdjMatrix
| | 2. T : Type
| | 3. T
```

```
| | 4. x : ℕM.size
| | 5. T→ℕM.size→T
| | 6. y : ℕM.size
| | 7. e : {p:(ℕM.size×ℕM.size)| M.adj(1of(p),2of(p)) }
| | 8. e = <x,y>
| | 9. (y ∈ upto(0;M.size))
| | ⊢  (y ∈ upto(0;M.size))
| |   by (RepeatFor 2 (ParallelOp -1)) THEN (Analyze -1)
| |
| |\
| | 1. M : AdjMatrix
| | 2. T : Type
| | 3. T
| | 4. x : ℕM.size
| | 5. T→ℕM.size→T
| | 6. y : ℕM.size
| | 7. e : {p:(ℕM.size×ℕM.size)| M.adj(1of(p),2of(p)) }
| | 8. e = <x,y>
| | ⊢  M.adj(x,y)
| | | by (Analyze -2) THEN Unhide THEN (Analyze -3) THEN (All Reduce)
| | |
| | 1. M : AdjMatrix
| | 2. T : Type
| | 3. T
| | 4. x : ℕM.size
| | 5. T→ℕM.size→T
| | 6. y : ℕM.size
| | 7. e1 : ℕM.size
| | 8. e2 : ℕM.size
| | 9. M.adj(e1,e2)
| | 10. <e1,e2> = <x,y>
| | ⊢  M.adj(x,y)  by Obvious
| |
| .\
| . 1. M : AdjMatrix
| . 2. T : Type
| . 3. T
| . 4. x : ℕM.size
| . 5. T→ℕM.size→T
| . 6. y : ℕM.size
| . 7. (y ∈ upto(0;M.size))
| . 8. M.adj(x,y)
| . ⊢  ∃e:{p:(ℕM.size×ℕM.size)| M.adj(1of(p),2of(p)) }. e = <x,y>
| .   by (InstConcl [<x,y>]) THEN Analyze THEN (Reduce 0)
|
.\
. 1. M : AdjMatrix
. 2. T : Type
. 3. s : T
. 4. x : ℕM.size
. 5. f : T→ℕM.size→T
. ⊢  primrec(M.size;s;λy,s'. if M.adj(x,y)→ f(s',y) else s' fi)
.    =
.    list_accum(s',x'.f(s',x');s;filter(M.adj(x);upto(0;M.size)))
.   by (RWO
```

```
.        Thm* ∀n:ℕ, f,x:Top.
.             primrec(n;x;f) ~ list_accum(i,y.f(y,i);x;upto(0;n))
.        {primrec_list_accum}
.        0)
.       THEN
.       (Reduce 0)
.       THEN
.       (RWO
.        Thm* ∀f:(T→A→T), P:(A→𝔹), L:A List, s:T.
.             list_accum(s',x'.f(s'
.                                  ,x');s;filter(P;L)) ~ ...
.        {list_accum_filter}
.        0)
END
```