# A Detailed Look at append's Reasoning Table

## append's Reasoning Table

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
S Code Assume Confirm
```

S	Code	Assume		Confirm
0		true		r0 * g0 = r0 * g0
1	<pre>while (g.length() &gt; 0) {   //! updates g, r   //! maintains   //! r * g = #r * #g   //! decreases  g </pre>	g1  > 0 ^ r1 * g1 = r0 * g0		
	Т у;			
2	4.	T.Init(y2)	Unchanged r, g	g2 /= <>
	g.dequeue(y);			
3		g3 = g2[1, g2 ) ^ <y3> = prefix of g2</y3>	Unchanged r	
	r.enqueue(y);			
4		T.Init(y4) ^ r4 = r3 * <y3></y3>	Unchanged g	g4  <  g1  ^ r4 * g4 = r0 * g0
	}			
5		~( g5  > 0) ^ r5 * g5 = r0 * g0		r5 = r0 * g0 ^ g5 = <>

## append's RT - Code

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
```

S	Code	Assume		Confirm
0		true		r0 * g0 = r0 * g0
	<pre>while(g.length() &gt; 0) { //! updates g, r //! maintains //! r * g = #r * #g //! decreases  g </pre>			
1		g1  > 0 r1 * g1 = r0		
	ту;			
2		T.Init(y2)	Unchanged r, g	g2 /= <>
	g.dequeue(y);			
3		g3 = g2[1, g2 ) ^ <y3> = prefix of g2</y3>	Unchanged r	
	r.enqueue(y);			
4		T.Init(y4) ^ r4 = r3 * <y3></y3>	Unchanged g	g4  <  g1  ^ r4 * g4 = r0 * g0
	}			
5		~( g5  > 0 r5 * g5 = r0		r5 = r0 * g0 ^ g5 = <>

#### append's RT - States

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
                Code
                                                                                     Confirm
                                                 Assume
                                                                               r0 * g0 = r0 * g0
                                                 true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                              |q1| > 0^{-}
                                          r1 * g1 = r0 * g0
       Ty;
                                                           Unchanged
                                                                                   g2 /= <>
                                      T.Init(y2)
                                                              r, g
       g.dequeue(y);
                                  g3 = g2[1, |g2|) ^
                                                           Unchanged
                                 <y3> = prefix of g2
                                                                r
       r.enqueue(y);
                                     T.Init(y4) ^
                                                           Unchanged
                                                                                 |g4| < |g1| ^
                                                                               r4 * g4 = r0 * g0
                                   r4 = r3 * < y3 >
                                            \sim (|g5| > 0) ^
                                                                                r5 = r0 * g0 ^
                                          r5 * g5 = r0 * g0
                                                                                    g5 = \langle \rangle
```

# append's RT – State 0 Assume

void	void appendVl (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver								
	//! updates r								
	/! clears g								
//!	//! requires: true								
//!	//! ensures: r = #r * #g								
	<b>\</b>								
S	Code	Assume		Confirm					
0		true		r0 * g0 = r0 * g0					
	<pre>while(g.length() &gt; 0) {</pre>								
	//! updates g, r								
	//! maintains								
	//! r * g = #r * #g								
	//! decreases  g								
1		g1  > 0							
		r1 * g1 = r0	* g0						
	T y;								
2		T.Init(y2)	Unchanged	g2 /= <>					
		1.11110(92)	r, g	92 /- 💔					
	g.dequeue(y);								
3		$g3 = g2[1,  g2 ) ^$	Unchanged						
		<y3> = prefix of g2</y3>	r						
	r.enqueue(y);								
4		T.Init(y4) ^	Unchanged	g4  <  g1  ^					
4		r4 = r3 * <y3></y3>	g	r4 * g4 = r0 * g0					
	}								
		~( g5  > 0		r5 = r0 * g0 ^					
5		r5 * g5 = r0	* g0	g5 = <>					
				30					

#### append's RT – State 0 Confirm

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                   Confirm
                                                Assume
                                                                            r0 * g0 = r0 * g0
                                                true
     while(g.length() > 0) {
     //! updates q. r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                             |g1| > 0^{\circ}
                                          r1 * g1 = r0 * g0
       Ty;
                                                          Unchanged
                                                                                  q2 /= <>
                                     T.Init(y2)
                                                             r, g
       g.dequeue(y);
                                 g3 = g2[1, |g2|) ^
                                                          Unchanged
                                <y3> = prefix of g2
                                                              r
       r.enqueue(y);
                                    T.Init(y4) ^
                                                          Unchanged
                                                                               |g4| < |g1| ^
                                                                             r4 * g4 = r0 * g0
                                   r4 = r3 * < y3 >
                                           \sim (|g5| > 0) ^
                                                                               r5 = r0 * g0 ^
                                         r5 * g5 = r0 * g0
                                                                                  q5 = <>
```

#### append's RT – State 1 Assume

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                     Confirm
                                                 Assume
                                                                              r0 * g0 = r0 * g0
                                                 true
     while (g.length() > 0) -(
     //! updates g, r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                           \rightarrow |q1| > 0^{\circ}
                                          r1 * g1 = r0 * g0
       Ty;
                                                           Unchanged
                                                                                   q2 /= <>
                                      T.Init(y2)
                                                              r, g
       g.dequeue(y);
                                  g3 = g2[1, |g2|) ^
                                                           Unchanged
                                 <y3> = prefix of g2
                                                               r
       r.enqueue(y);
                                    T.Init(y4) ^
                                                           Unchanged
                                                                                 |g4| < |g1| ^
                                                                              r4 * g4 = r0 * g0
                                   r4 = r3 * < y3 >
                                            \sim (|g5| > 0) ^
                                                                                r5 = r0 * g0 ^
                                          r5 * g5 = r0 * g0
                                                                                    q5 = <>
```

#### append's RT – State 1 Assume

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                   Confirm
                                                Assume
                                                                             r0 * g0 = r0 * g0
                                                true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * g = #r * #g --
     //! decreases |g|
                                             |g1| > 0^{\circ}
                                        r1 * g1 = r0 * g0 
       Ty;
                                                          Unchanged
                                                                                  q2 /= <>
                                     T.Init(y2)
                                                             r, g
       g.dequeue(y);
                                 g3 = g2[1, |g2|) ^
                                                          Unchanged
                                <y3> = prefix of g2
                                                              r
       r.enqueue(y);
                                    T.Init(y4) ^
                                                          Unchanged
                                                                               |g4| < |g1| ^
                                                                             r4 * g4 = r0 * g0
                                   r4 = r3 * < y3 >
                                           \sim (|g5| > 0) ^
                                                                               r5 = r0 * g0 ^
                                         r5 * g5 = r0 * g0
                                                                                  q5 = <>
```

#### append's RT – State 1 Confirm

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                   Confirm
                                                Assume
                                                                             r0 * g0 = r0 * g0
                                                true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                             |q1| > 0^{\circ}
                                                                                  true
                                         r1 * g1 = r0 * g0
       Ty;
                                                          Unchanged
                                     T.Init(y2)
                                                                                 g2 /= <>
                                                            r, g
       g.dequeue(y);
                                 g3 = g2[1, |g2|) ^
                                                          Unchanged
                                <y3> = prefix of g2
                                                              r
       r.enqueue(y);
                                    T.Init(y4) ^
                                                          Unchanged
                                                                               |g4| < |g1| ^
                                                                             r4 * g4 = r0 * g0
                                   r4 = r3 * < y3 >
                                           \sim (|g5| > 0) ^
                                                                              r5 = r0 * g0 ^
                                         r5 * g5 = r0 * g0
                                                                                  q5 = <>
```

#### append's RT – State 2 Assume

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                   Confirm
                                                Assume
                                                                             r0 * g0 = r0 * g0
                                                true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                             |g1| > 0^{\circ}
                                         r1 * g1 = r0 * g0
       Ty;
                                                          Unchanged
                                 T.Init(y2)
                                                                                 q2 /= <>
                                                            r, g
       g.dequeue(y);
                                 g3 = g2[1, |g2|) ^
                                                          Unchanged
                                <y3> = prefix of g2
                                                              r
       r.enqueue(y);
                                    T.Init(y4) ^
                                                          Unchanged
                                                                               |g4| < |g1| ^
                                  r4 = r3 * < y3 >
                                                                             r4 * g4 = r0 * g0
                                           \sim (|g5| > 0) ^
                                                                              r5 = r0 * g0 ^
                                         r5 * g5 = r0 * g0
                                                                                  q5 = <>
```

#### append's RT – State 2 Assume

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                   Confirm
                                                Assume
                                                                             r0 * g0 = r0 * g0
                                                true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                             |g1| > 0^{\circ}
                                          r1 * g1 = r0 * g0
       Ty;
                                                          Unchanged
                                     T.Init(y2)
                                                                                  q2 /= <>
                                                             r, g
       g.dequeue(y);
                                 g3 = g2[1, |g2|) ^
                                                          Unchanged
                                <y3> = prefix of g2
       r.enqueue(y);
                                    T.Init(y4) ^
                                                          Unchanged
                                                                               |g4| < |g1| ^
                                   r4 = r3 * < y3 >
                                                                             r4 * g4 = r0 * g0
                                           \sim (|g5| > 0) ^
                                                                               r5 = r0 * g0 ^
                                         r5 * g5 = r0 * g0
                                                                                  q5 = <>
```

# append's RT – State 2 Assume

	<pre>void appendV1 (QueueOfT&amp; r, QueueOfT&amp; g) // Using r for receiver, g for giver</pre>								
	//! updates r								
	//! clears g								
	//! requires: true								
//!	//! ensures: r = #r * #g								
S	Code	Assume		Confirm					
0		true		r0 * g0 = r0 * g0					
		5140		10 g0 = 10 g0					
	<pre>while(g.length() &gt; 0) {</pre>								
	//! updates g, r			m2 - m1					
	//! maintains			r2 = r1					
	//! r * g = #r * #g			g2 = g1					
	//! decreases  g								
1		g1  > 0							
		r1 * g1 = r0	* g0						
	T y;								
2		T.Init(y2)	Unchanged	g2 /= <>					
		1.1HIC (y2)	r, g	g2 /- <>					
	g.dequeue(y);								
3		$g3 = g2[1,  g2 ) ^$	Unchanged						
3		<y3 $>$ = prefix of g2	r						
	r.enqueue(y);								
4		T.Init(y4) ^	Unchanged	g4  <  g1  ^					
4		r4 = r3 * < y3 >	g	r4 * g4 = r0 * g0					
	}								
		~( g5  > 0		r5 = r0 * g0 ^					
5		r5 * g5 = r0	* g0	g5 = <>					
				95 - \>					

#### append's RT – State 2 Confirm

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                   Confirm
                                                Assume
                                                                            r0 * g0 = r0 * g0
                                                true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                             |g1| > 0^{\circ}
                                         r1 * g1 = r0 * g0
       Ty;
                                                          Unchanged
                                                                              g2 /= <>
                                     T.Init(y2)
                                                            r, g
       g.dequeue(y);
                                 g3 = g2[1, |g2|) ^
                                                          Unchanged
                                <y3> = prefix of g2
                                                              r
       r.enqueue(y);
                                    T.Init(y4) ^
                                                          Unchanged
                                                                               |g4| < |g1| ^
                                                                            r4 * g4 = r0 * g0
                                  r4 = r3 * < y3 >
                                           \sim (|g5| > 0) ^
                                                                              r5 = r0 * g0 ^
                                         r5 * q5 = r0 * q0
                                                                                  q5 = <>
```

#### append's RT – State 3 Assume

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                     Confirm
                                                 Assume
                                                                               r0 * g0 = r0 * g0
                                                 true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                              |g1| > 0^{\circ}
                                          r1 * g1 = r0 * g0
       Ty;
                                                           Unchanged
                                                                                    q2 /= <>
                                      T.Init(y2)
                                                              r, g
       g.dequeue(y); =
                               g3 = g2[1, |g2|) ^
                                                           Unchanged
                                \langle y3 \rangle = prefix of q2
                                                                r
       r.enqueue(y);
                                     T.Init(y4) ^
                                                           Unchanged
                                                                                 |g4| < |g1| ^
                                                                               r4 * g4 = r0 * g0
                                   r4 = r3 * < y3 >
                                            \sim (|g5| > 0) ^
                                                                                r5 = r0 * g0 ^
                                          r5 * g5 = r0 * g0
                                                                                    q5 = <>
```

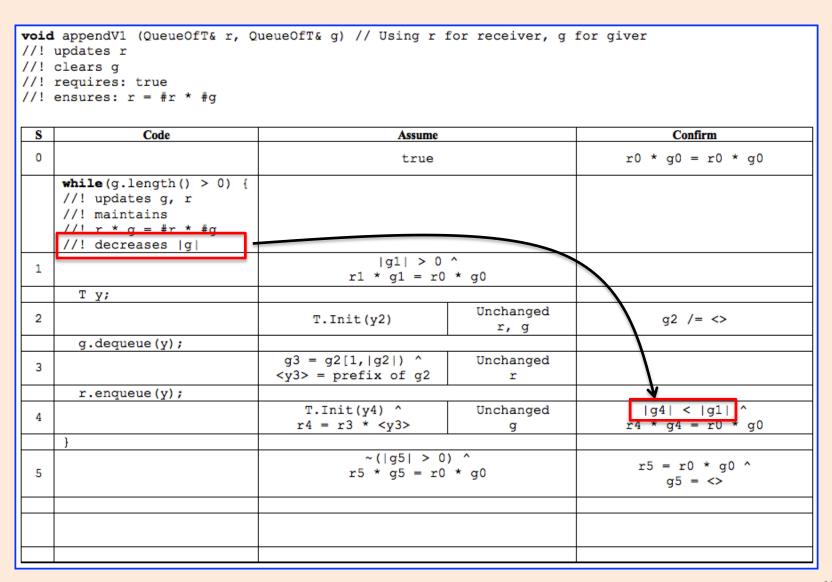
#### append's RT – State 3 Confirm

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
                Code
                                                                                     Confirm
                                                 Assume
                                                                               r0 * g0 = r0 * g0
                                                 true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                              |g1| > 0^{\circ}
                                          r1 * g1 = r0 * g0
       Ty;
                                                           Unchanged
                                                                                    g2 /= <>
                                      T.Init(y2)
                                                              r, g
       g.dequeue(y);
                                  q3 = g2[1, |g2|) ^
                                                           Unchanged
                                                                                     true
                                 \langle y3 \rangle = prefix of q2
       r.enqueue(y);
                                     T.Init(y4) ^
                                                           Unchanged
                                                                                 |g4| < |g1| ^
                                                                               r4 * g4 = r0 * g0
                                    r4 = r3 * < y3 >
                                            \sim (|g5| > 0) ^
                                                                                r5 = r0 * g0 ^
                                          r5 * g5 = r0 * g0
                                                                                    q5 = <>
```

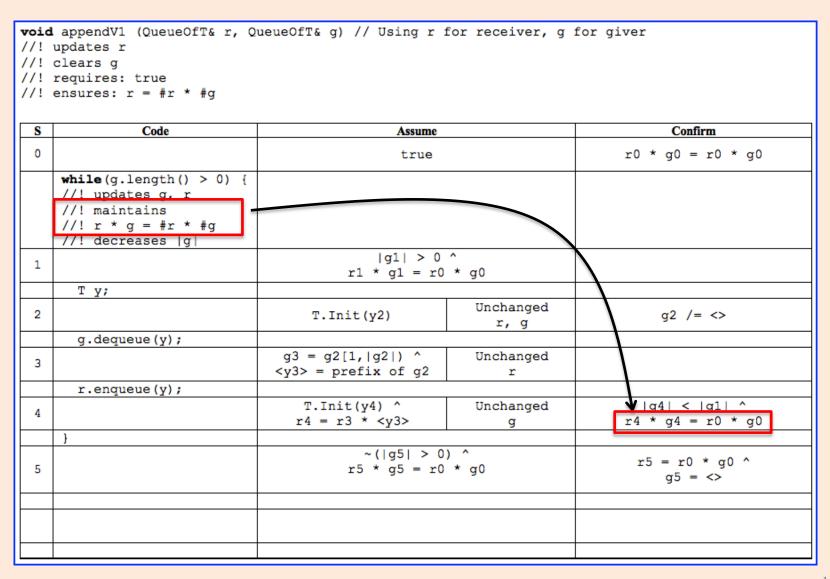
#### append's RT – State 4 Assume

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                     Confirm
                                                 Assume
                                                                              r0 * g0 = r0 * g0
                                                 true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                              |g1| > 0^{\circ}
                                          r1 * g1 = r0 * g0
       Ty;
                                                           Unchanged
                                                                                   q2 /= <>
                                      T.Init(y2)
                                                              r, g
       g.dequeue(y);
                                  g3 = g2[1, |g2|) ^
                                                           Unchanged
                                 <y3> = prefix of g2
                                                               r
       r.enqueue(y);=
                                 T.Init(y4) ^
r4 = r3 * <y3>
                                                           Unchanged
                                                                                 |g4| < |g1| ^
                                                                              r4 * g4 = r0 * g0
                                            \sim (|g5| > 0) ^
                                                                                r5 = r0 * g0 ^
                                          r5 * g5 = r0 * g0
                                                                                    q5 = <>
```

#### append's RT – State 4 Confirm



#### append's RT – State 4 Confirm



#### append's RT – State 5 Assume

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                     Confirm
                                                 Assume
                                                                               r0 * g0 = r0 * g0
                                                 true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * g = #r * #g
     //! decreases |g|
                                              |g1| > 0^{\circ}
                                          r1 * g1 = r0 * g0
       Ty;
                                                           Unchanged
                                                                                   q2 /= <>
                                      T.Init(y2)
                                                              r, g
       g.dequeue(y);
                                  g3 = g2[1, |g2|) ^
                                                           Unchanged
                                 <y3> = prefix of g2
                                                                r
       r.enqueue(y);
                                     T.Init(y4) ^
                                                           Unchanged
                                                                                 |g4| < |g1| ^
                                                                               r4 * g4 = r0 * g0
                                    r4 = r3 * < y3 >
                                          ~(|g5| > 0) ^
r5 * g5 = r0 * g0
                                                                                r5 = r0 * g0 ^
                                                                                    q5 = <>
```

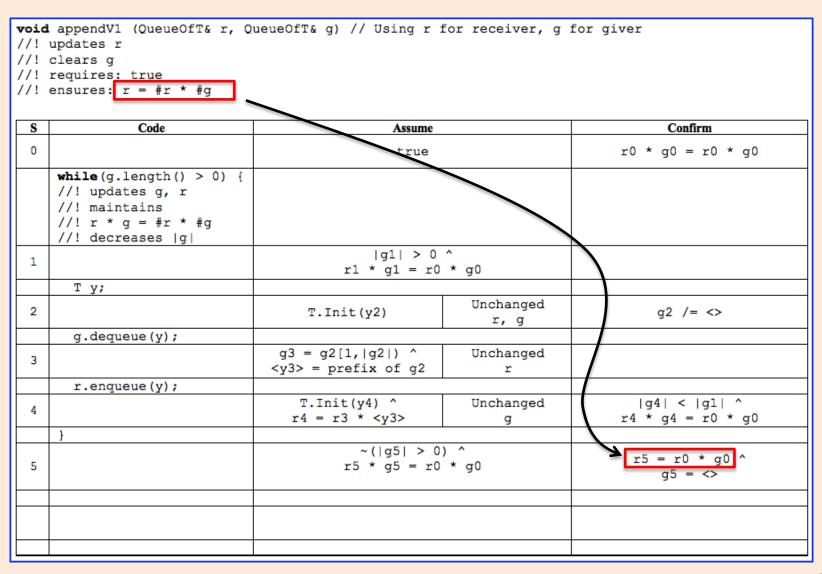
# append's RT – State 5 Assume

	roid appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver								
	//! updates r								
	/! clears g								
	/! requires: true								
//:	//! ensures: r = #r * #g								
S	Code		Assume		Confirm				
0			true		r0 * g0 = r0 * g0				
	<pre>while (g.length() &gt; ( //! updates g, r</pre>	} {							
	//! maintains								
	//! r * g = #r * #g								
	//! decreases  g								
1			g1  > 0						
_			r1 * g1 = r0	* g0					
	T y;								
2			T.Init(y2)	Unchanged	g2 /= <>				
			111110 (32)	r, g	92 / 4				
	g.dequeue(y);								
3			$g3 = g2[1,  g2 ) ^$	Unchanged					
			<y3 $>$ = prefix of g2	r					
	r.enqueue(y);								
4		<b>\</b>	T.Init(y4) ^	Unchanged	g4  <  g1  ^				
		1	r4 = r3 * <y3></y3>	g	r4 * g4 = r0 * g0				
	}	1		\ <u>*</u>					
5			~( g5  > 0 r5 * g5 = r0	* g0	r5 = r0 * g0 ^ g5 = <>				
				7					
			g5  = 0						

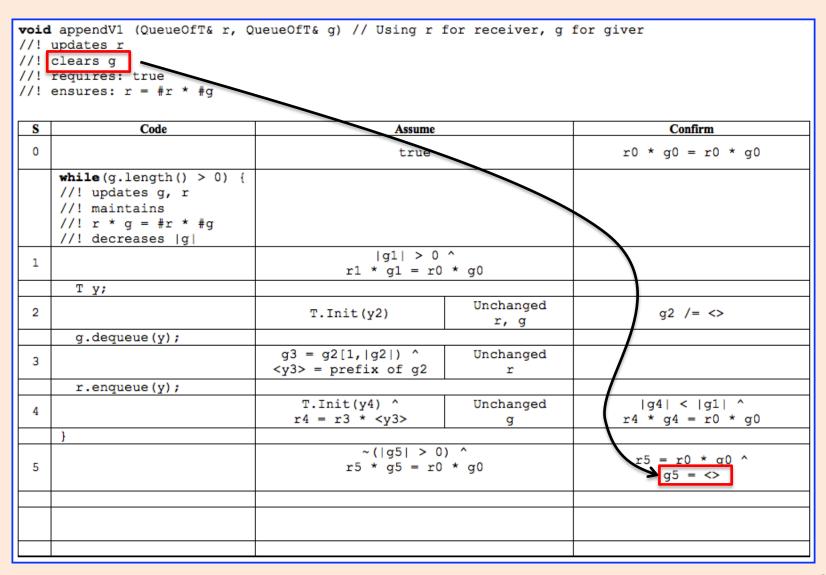
#### append's RT – State 5 Assume

```
void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g
               Code
                                                                                    Confirm
                                                 Assume
                                                                              r0 * q0 = r0 * q0
                                                 true
     while (g.length() > 0) {
     //! updates g, r
     //! maintains
     //! r * q = #r * #q
     //! decreases |g|
                                              |q1| > 0^{\circ}
 1
                                          r1 * g1 = r0 * g0
       Ty;
                                                           Unchanged
                                                                                   q2 /= <>
                                     T.Init(y2)
                                                             r, g
       g.dequeue(y);
                                 g3 = g2[1, |g2|) ^
                                                           Unchanged
                                 <y3> = prefix of g2
                                                               r
       r.enqueue(y);
                                    T.Init(y4) ^
                                                                                |g4| < |g1| ^
                                                           Unchanged
                                                                              r4 * q4 = r0 * q0
                                   r4 = r3 * < y3 >
                                            \sim (|\alpha 5| > 0) ^
                                                                               r5 = r0 * g0 ^
                                          r5 * g5 = r0 * g0
                                                                                   g5 = <>
```

#### append's RT – State 5 Confirm



#### append's RT – State 5 Confirm



## Loop Invariant Requires Confirming

In locations 1 and 2 is where we must prove the loop invariant holds

S	Code	Assume		Confirm	
0	(1)	true		r0 * g0 = r0 * g0 ①	
	<pre>while (g.length() &gt; 0) { //! updates g, r //! maintains //! r * g = #r * #g //! decreases  g </pre>				<u></u>
1		gl  > 0 rl * gl = r0		32 × 50	(\bullet)
2	T y;	T.Init(y2)	Unchanged r, g	g2 /= <>	test 3
0	g.dequeue(y);				I (X) ***
3	5281802018B	g3 = g2[1, g2 ) ^ <y3> = prefix of g2</y3>	Unchanged r	/	true
	r.enqueue(y);			J.	
4	25	T.Init(y4) ^ r4 = r3 * <y3></y3>	Unchanged g	g4  <  g1  ^ r4 * g4 = r0 * g0	loop-body
	1				
5		~( g5  > 0) r5 * g5 = r0 '		r5 = r0 * g0 ^ g5 = <>	
7					•

## Loop Invariant Assumed True

In locations 3 and 4 is where we get to assume the loop invariant holds

S	Code	Assume		Confirm	
0		true		r0 * g0 = r0 * g0	
	<pre>while (g.length() &gt; 0) { //! updates g, r //! maintains //! r * g = #r * #g //! decreases  g </pre>				<u></u>
1	Control of the Contro	g1  > 0 rl * g1 = r0 '			
2	Т у;	T.Init(y2)	Unchanged r, g	g2 /= <>	test 3
	g.dequeue(y);	3		100	
3	11111111	g3 = g2[1, g2 ) ^ <y3> = prefix of g2</y3>	Unchanged r		true
	r.enqueue(y);		111 200		
4	1	T.Init(y4) ^ r4 = r3 * <y3></y3>	Unchanged g	g4  <  g1  ^ r4 * g4 = r0 * g0	loop-body
	1	555			
5		~( g5  > 0) r5 * g5 = r0		r5 = r0 * g0 ^ g5 = <>	
7			1		'ノ

# append's RT – Major Assertions

and A constant (Occurs Office on Occurs Office on ) // Walter or for one of the occurs								
<pre>void appendVl (QueueOfT&amp; r, QueueOfT&amp; g) // Using r for receiver, g for giver //! updates r</pre>								
//!_clears g								
//! requires: true								
/   ensures: r = #r * #g								
Code	Code Assume Confirm							
<b>F</b> /	true		r0 * g0 = r0 * g0					
<pre>while(g.length() &gt; 0) {</pre>								
//! updates g, r								
//! maintains								
//! r * g = #r * #g //! decreases  g								
//: decreases [g]	g1  > 0	^						
	rl * gl = r0 *							
T y;								
2	T.Init(y2)	Unchan red r, g	g2 /= <>					
g.dequeue(y);								
IXI	g3 = g2[1, g2 ) ^ (y3> = prefix of g2	Unchanged r						
r.enqueue(y);	1	1						
4	T.Init(y4) ^ r4 = r3 * <y3></y3>	Unchanged g	$ g4  <  g1  ^{2}$ $r4 * g4 = r0 * g0^{2}$					
<del>                                     </del>			2. 9. 20 90					
	~( g5  > 0)	^						
5	r5 * g5 = r0 *	g0 <mark></mark> ①	r5 = r0 * g0 ^ g5 = <>					
7								
	· · · · · · · · · · · · · · · · · · ·							

### Up Next – append's VCs

- 1. Using the assertions in this RT to generate append's VCs
- 2. Proving the VCs to prove append is correct, i.e., meets its spec